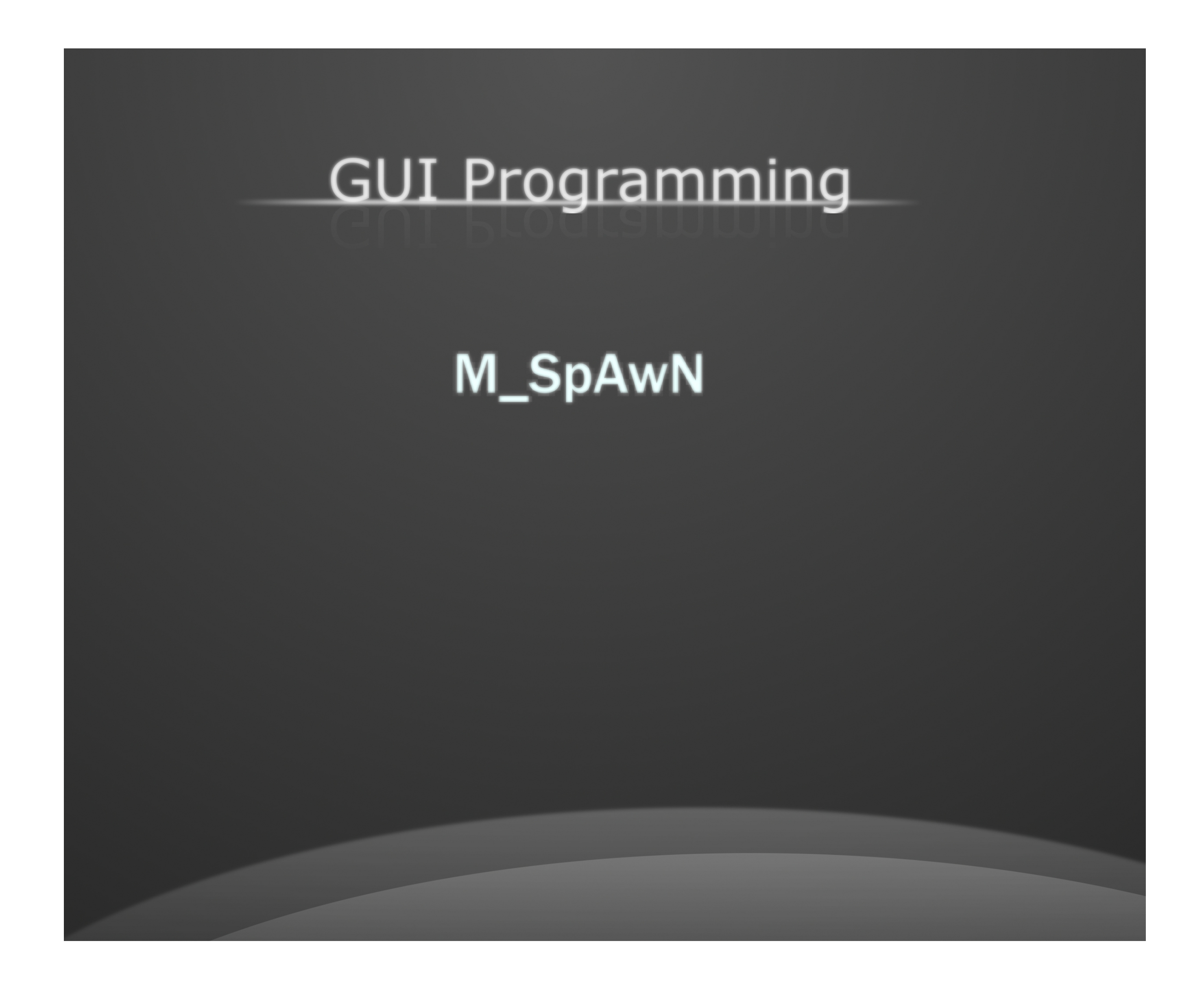

GUI Programming
M_SpAwN

Perl Gui Programming Book

Introduction

في هذا الكتاب سوف نتطرق الى برمجة الواجهات الرسومية في لغة البيرل من خلال استعمال الموديل الخاص بهذه العملية وهو الموديل المعروف بأسم ال(*Tk*)
هذا الموديل متوفر على شبكة ال

www.cpan.org

وهي الشبكة المتخصصة في نشر كل الموديلات الخاصة في لغة البيرل و الان بعد ان قمت بتحميل الموديل من الانترنت ما عليك سوى ان تنصبه من خلال خطوات تنصيب الموديل وذلك كما يلي بعد ان تفتح كبس الموديلوبعد ذلك تقوم بتنصيبه على جهازك وكما يلي

1-perl Makefile.PL
2-make
3-make install

ومن بعد ان تقوم بهذه الخطوات سوف يتم تنصيب الموديل لديك وتقدر ان تبرمج به كما تحب
تسمية الموديل(*Tk*)
على حسب حسب ما وجدت على الانترنت انها مختصر لكلمة
Tool Kit
كما ايضا لاحظت في الاعداد السابقة لم اعطي الاسم المختصر للغة البيرل لا ادري لماذا ولكن على كل كلمة بيرل هي مختصر لكلمة
Practical Extraction and Report Language
اما عن برمجة الواجهات الرسومية كما يعلم الكل فأنها مختصر لكلمة
Graphical User Interface

واما عن هذا الكتاب فأنه يحتوي على اغلب المكونات التي تأتي مع الموديل(*Tk*)
وعندما قمت بكتابة الكتاب قررت انه بعد الانتهاء من الكتاب راح انزل ان شاء الله ملحق يضم ال
Very advanced methods of Tk programming
وعلى طول فترة قرأتك للكتاب لن تلاحظ على ما أظن وجود شئ يصعب عليكم لانه اغلب البرامج التي تم استعمالها هي برامج سهلة وتم شرحها شرح مفصل مع الصور الناتجة عن تنفيذ هذه البرامج

LabelS

الخطوة الاولى و البرنامج الاول من برمجة الواجهات الرسومية
البرنامج الاول

***CODE(1)**

```
use Tk;                                          #1
$top=MainWindow->new;                            #2
$a=$top->Label( -text      =>"Hello world"        #3
          );                                     #4
$a->pack;                                        #5
MainLoop;                                        #6
```

الان سوف نأتي على على شرح هذا الكود
لكي نعرف ما هي المبادئ الاساسية التي يعتمد عليها البرنامج لكي يتم عرضه بهذا الصورة
الان سوف نأتي الى تسلسل الخطوات
الخطوة الاولى

***CODE(2)**

```
use Tk;
```

الان نلاحظ ان هذه هي الخطوة الاولى التي من خلالها يتم استيراد التعاريف الخاصة و الروتينات الفرعية و استيراد كل ما يتعلق بعمل هذا الموديل الرسومي
شئ مهم تجدر الاشارة اليه هي انه هذه الموديلات التي يتم كتابتها بالشكل الذي كتبت فيه اي انه الحرف الاول منها هو حرف ذو مستوى كبير
وان اي تغيير فيه هو سوف يؤدي الى حصول اخطاء كثيرة وبالتالي الى عدم تنفيذ البرنامج

الخطوة الثانية

***CODE(3)**

```
$top=MainWindow->new;
```

هذه كما هو ملاحظ انها الخطوة الثانية من خطوات برمجة البرنامج على كل ان عمل هذه الخطوة هو انه تعمل على تصميم النافذة الاساسية الكبيرة التي يتم بداخلها عرض ما قد قمت ببرمجته لكي يتم عرضه
ومن الملاحظ ان هذا السطر يحتوي على متغير يعمل على اسناد قيمة للقيمة المسئولة عن عرض الشاشة الكبيرة الخاصة في البرنامج
ملاحظة
يرجى ملاحظة ان كلمة(*MainWindow*)
كل من الحرفان الاولان من كل كلمة هو حرف كبير لذا يرجى الانتباه
الخطوة الثالثة

***CODE(4)**

```
$a=$top->Label ( -text => "Hello world"
                 );
```

هذا السطر يعمل على وضع اول(*widget*)
في البرنامج الاساسي وذلك بناء على طلب من ال (لابل)

والذي يحتوي على ما يسمى بالشاهد الذي يحتوي على خيار ال(*text*)
والنص الذي ترغب في ان يتم عرضه على النافذة الاساسية للبرنامج ولكن لاحظ اننا نقول اننا نعرض ولا نطبع
لانه لو تم طباعته سوف ما كانت سوف تتم عملية العرض لانه ايضا نلاحظ على طول البرنامج انه لاتوجد كلمة
اطبع
وان خيار ال *text*)
الذي اسندت اليه ان يتم عرض كلمة
(*hello world*)
ومن الممكن ان يتم اسناد اي قيمة اخرى غير تلك القيمة

الخطوة الخامسة

CODE(5)

```
$a->pack;
```

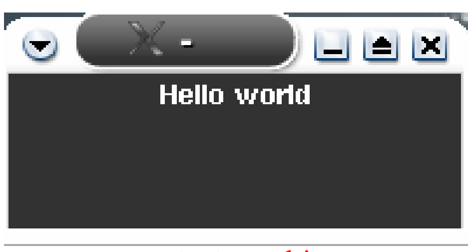
هذا السطر يعمل على حزم الامور وانه يعمل على اخبار مدير هندسة الامر(pack)
ان يدير المتغير و يعمل على حزمه استعدادا لكي يتم عرضه في نافذة جديدة

الخطوة السادسة

CODE(6)

```
MainLoop;
```

هذا الخطوة تكون مسئولة
والان بعد ان تمت كاتبة البرنامج كله كان لابد من ان يوجد خطوة تعمل على اخذ جميع هذه الخطوات البرمجية
السابقة وان تعمل على ان يتم ادخالها في حبكة برمجية واحدة لكي يفهم المترجم الخاص بلغة البيرل ما هو
المطلوب منه و الذي هو سوف يكون بصدد عرضه
والان بعد ان انتهينا من كاتبة البرنامج وفهمنا ما هو عمل جميع هذه الخطوات سوف نقوم بتنفيذ البرنامج لكي
نرى ماهو ناتج التنفيذ والناتج موضح في الصورة التالية



FIGURE(1)

الان كما تلاحظ انه هذا البرنامج الذي قمنا بكتابته قبل قليل و هذا هو ناتج التنفيذ الخاص به

How to create a perl Program with title name

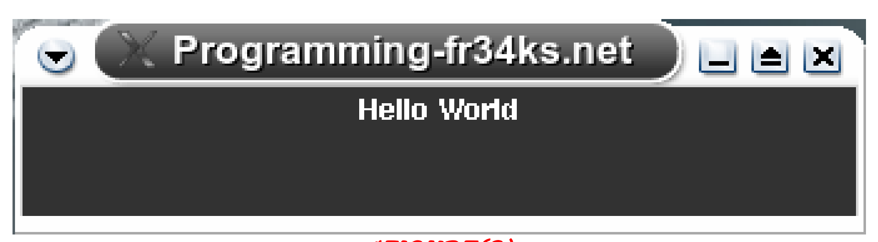
الان كما نلاحظ في البرنامج السابق الذي سبق ان تم عرضه نلاحظ في الشريط الاعلى لهذا البرنامج انه توجد صورة لشكل يشبه حرف الاكس في اللغة الانكليزية الى جانبها علامة هي علامة الشرطة او علامة ال -

الان لو كنت ترغب ان يتم ان يحمل برنامجك اسم ويكون له اسم فان العملية و البرنامج سوف يكون كما يلي في هذا الكود

*CODE(7)

```
use Tk;
$top=MainWindow->new;
$a=$top->Label( -text => "Hello World"
                );
$top->title("Programming-fr34ks.net");
$a->pack;
MainLoop;
```

والان البرنامج هذا تماما هو نفس البرنامج السابق ولكن توجد فيه نقطة اختلاف واحدة هي تكمن في الخيار الذي يكون مسئول عن وضع الاسم الخاص في البرنامج وكما هو ملاحظ انه هذا الخيار هو(title)
ومن ثم نلحقه باسم البرنامج وهكذا تتم العملية الان عند تنفيذ هذا البرنامج سوف يكون عرض البرنامج بالصورة الاتية وكما يلي



*FIGURE(2)

الان نلاحظ بعد ان تم تنفيذ البرنامج بشكل كامل نلاحظ انه الان اصبح يحمل الاسم الذي قمنا باطلاقه عليه حيث يمكن القول انه خطوة بسيطة ومن الجميل ان نطلق على كل برنامج من هذا النوع اسم خاص به

Some advanced features on label

الان سوف نتناول بعض الخواص المتقدمة التي تعود الى ال(*Label*)
الخاصية الاولى
و هي الخاصية الخاصة بالتلوين وان التلوين يكون على نوعين الان سوف نأخذ الخاصية الاولى من التلوين لكي نرى ماهو ناتج التنفيذ كما في هذا الكود الاتي

CODE(8)

```
use Tk;
$top=MainWindow->new;
$a=$top->Label( -text => "Hello World",
                -background   => "red"
               );
$top->title("Programming-fr34ks");
$a->pack;
MainLoop;
```

الان نلاحظ ان ناتج تنفيذ هذا الكود هو كما في هذه الصورة الاتية



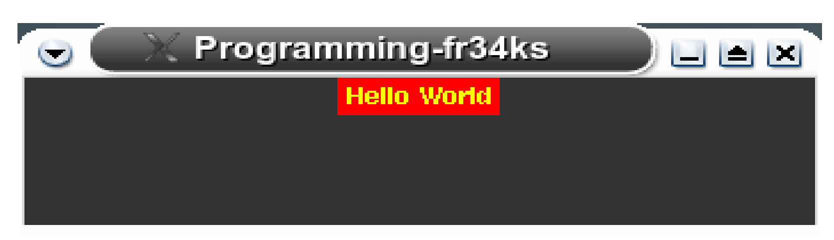
FIGURE(3)

الان نلاحظ ان تلوين الخلفية الخاصة بهذا الامر تم تلوينها باللون الاحمر الذي قمنا بتحديده ومن الممكن ان يتم تغير الالوان الى الالوان اخرى مثل اخضر او ازرق او اي لون اخر انت تحبه
الخاصية الثانية
هنالك خاصية اخرى من الممكن ان يتم استعمالها لكي يتم اضافة خاصية اخر الى امكانيات البرنامج و هذه الخاصية خاصية ال(*foreground*)
ويتم استعمالها كما في هذا الكود الاتي

CODE(9)

```
use Tk;
$top=MainWindow->new;
$a=$top->Label( -text => "Hello World",
                -background   => "red",
                -foreground   =>"yellow"
               );
$top->title("Programming-fr34ks");
$a->pack;
MainLoop;
```

الان نلاحظ انه عندما يتم تنفيذ هذا الكود سوف نلاحظ حدوث ما يلي فيه كما في الصورة الاتية



*FIGURE(4)

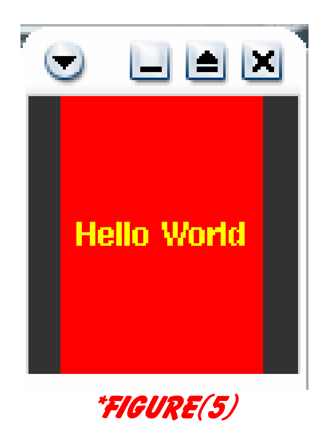
الخاصية الثالثة

ومن الخواص الاخرى التي تكون متعلقة بهذه الخيار هي خيارات المتعلقة بالاحداثيات الطول و العرض ومن الممكن ان يتم دمجها مع الخيارات الماضية كما في هذا الكود الاتي

*CODE(10)

```
use Tk;
$top=MainWindow->new;
$a=$top->Label( -text => "Hello World",
                -background   => "red",
                -foreground   =>"yellow",
                -height       =>7.5,
                -width        =>10
                );
$top->title("Programming-fr34ks");
$a->pack;
MainLoop;
```

الان عند تنفيذ هذا الكود سوف نحصل على ما يلي كما في هذه الصورة ولكن على كل الصورة التي راح نحصل عليها هي صورة غير منتظمة لانني لم اضبط الاعدادات الخاصة بها على كل الصورة هي كما يلي



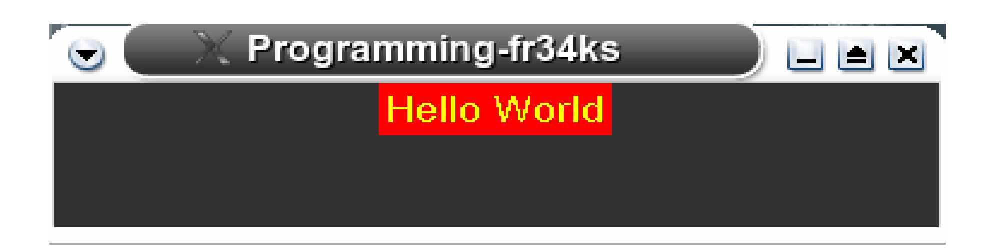
*FIGURE(5)

الخاصية الرابعة
ومن الخواص الاخرى الموجودة لدى هذا الامر هي خاصية الفونت اي انه نقوم بتحديد الخط الذي نريد ان نكتب به وتتم العملية كما يلي في هذا الكود

*CODE(11)

```
use Tk;
$top=MainWindow->new;
$a=$top->Label( -text => "Hello World",
                -background   => "red",
                -foreground   =>"yellow",
                -font         =>"arial"
                );
$top->title("Programming-fr34ks");
$a->pack;
MainLoop;
```

الان عند تنفيذ هذا الكود سوف نحصل على هذه الصورة الناتجة من التنفيذ ومن الممكن مباشرة ان نلاحظ الفرق بين هذه الصورة والفرق بين الصورة التي مضت من ناحية الخط



*FIGURE(6)

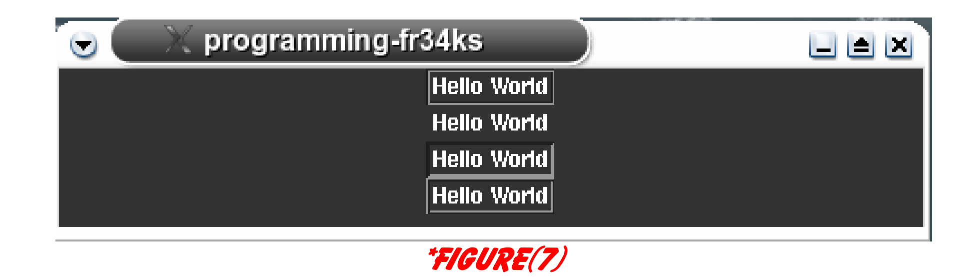
الخاصية الخامسة
الان هذه الخاصية هي تكون مسئولة عن تكون الزخرفة الخاصة المكونة للكلمة التي تم اسنادها الى الخيار (label)
وهذه الخاصية هي خاصية اسمها ال(relief)
والان عند تنفيذ كود قد تم ادخال فيه هذا الخيار سوف نحصل على ما يلي
والان سوف نقوم بكتابة كود هذا الكود يحتوي على كافة الخواص المتعلقة بهذا الخيار اي الخيار(relief)
واليكم الكود الخاص بهذه العملية وسوف نلاحظ الفرق بين هذه بين كل كلمة و الزخرفة التي تحيط بها الكود

***CODE(12)**

```
use Tk;
$top=MainWindow->new;
$top->title("programming-fr34ks");
$a=$top->Label (     -text          =>"Hello World",
                     -relief        =>"groove"
);
$a->pack;
$b=$top->Label(           -text    =>"Hello World",
                         -relief   =>"flat"
);
$b->pack;


$c=$top->Label(           -text    =>"Hello World",
                         -relief   =>"raised"
);
$d=$top->Label(           -text    =>"Hello World",
                         -relief   =>"sunken"
);
$d->pack;
$e=$top->Label(           -text    =>"Hello World",
                          -relief  =>"solid"
);
$f=$top->Label(           -text    =>"Hello World",
                         -relief   =>"ridge"
);
$f->pack;
MainLoop;
```

الان عند تنفيذ هذا الكود سوف نحصل على هذه الصورة كناتج على نجاح التنفيذ



***FIGURE(7)**

Buttons

الان سوف نتكلم عن ما يخص الازرار وما يتعلق بها وما يتعلق ببرمجتها وكيف يتم انشأها وخياراتها و الخ من الامور المتعلقة بها

اولا وقبل البداية الكل يعرف انه الازرار في لغة البيرل وفي كل اللغات هي تكون على نوعين

1-check buttons
2-radio buttons

الان سوف نقوم بكتابة كود من شأنه ان يقوم بأنشاء زر وسوف نتكلم اكثر بعد الكود

CODE(13)

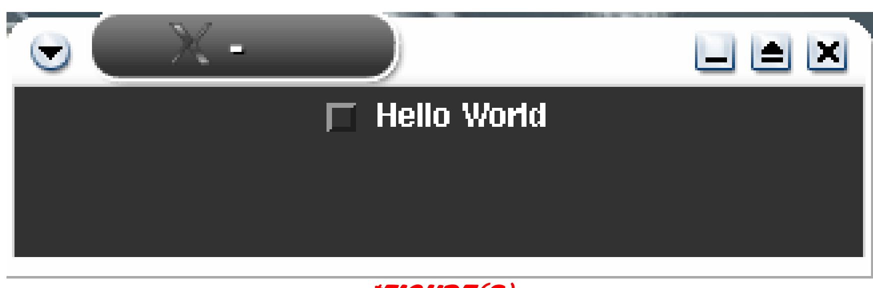
```
use Tk;
$top=MainWindow->new;
$a=$top->Checkbutton (-text =>"Hello World");
$a->pack;
MainLoop;
```

الان من خلال نظرة مبدئية على هذا الكود سوف نلاحظ انه يشبه الكودات الاخرى التي سبق ان قمنا بكتابتها في الصفحات الماضية ولكن فقط وجهة الاخلاف الوحيدة من الناحية الكتابية ان امكن القول

هو انه هذا الكود يحتوي على كلمة(*Checkbuttons*)

والكودات الاخرى الماضية تحتوي على كلمة(*Label*)

والان نلاحظ انه هذا الكود على العموم ليس بمقدار من الصعوبة و انه لا يحتاج الى شرح و الان اذا قمنا بتنفيذ هذا الكود سوف نحصل على نتيجة عرض كما في الصورة الاتية



FIGURE(8)

الان كما نلاحظ انه لاحظنا نتيجة تنفيذ الكود طبعا اكيد ومن الممكن ان نعمل على اضافة الخواص البرمجية التي ذكرناها في الكودات السابقة مثل الفونت و الخلفية الخاصة باللون و خلفية الكلمات

يعني الباك كراوند و الفوركراوند على التوالي ولكن هنالك فقرة لم اتطرق عليها في الموضوع السابق وهي انه من اللمكن ان تتم اضافة اوامر تقوم هذه الاوامر باعمال معينة و الان سوف نكتب الكود الخاص بما اعنيه لكي نتناقش في ما يعني ذلك

CODE(14)

```
use Tk;
$top=MainWindow->new;
$a=$top->Checkbutton (     -text            =>"Hello world",
                           -command     =>\&exit
);
$a->pack;
MainLoop;
```

الان نلاحظ هذا الكود انه يحتوي على فقرة واحدة تختلف عن الكودات السابقة وهذه الفقرة هي

CODE(15)

```
        -command     =>\&exit
```

هذه هو المقطع الغريب الذي كنت اتحدث عليه والغريب فيه هو انه ليس ظاهريا!!!
الان اكيد سوف يسأل احد ويقول نحن نشتغل على ال(Gui programming)
اذن ما فائدته اذا لم تكن ظاهريا اذن هذه فقرة سوف نتعلمها هو انه ليس كل ما يكتب في داخل الكود سوف تكون قادرا على رؤية ما يحتويه في البرنامج بعد عرضة الان نرجع الى فقرة هذا السطر البرمجي هو انه عندما ننفذ الكود ونأتي الى التشيك بوتون الى قد قمنا في عملها وعندما نقوم بالكبس عليها فلن نحصل على تأشير الذي عهدناه من الحالات السابقة ولكن الذي سوف يحصل هو انه عند الكبس عليها سوف يتم الخروج من البرنامج هذه هي اهمية هذه الخطوة
اي الان ان امكن تعريفها يمكن ان نعرفها كما يلي
الكوماند تحدد رفرنس الى روتين فرعي في البيرل يؤدي وظيفة معينة عند الكبس او الضغط على الزر الذي يكون فيه مفعل

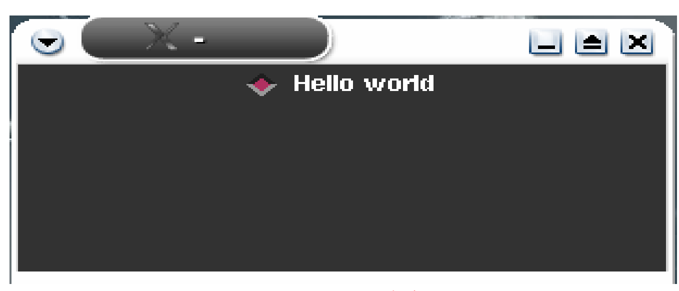
Radio Button

الان بعد ان عرفنا ماهي التشيك بوتن سوف نصمم الراديو بوتن والعملية تتم كما يلي في هذا الكود

*CODE(16)

```
use Tk;
$top=MainWindow->new;
$a=$top->Radiobutton (      -text              =>"Hello world"
);
$a->pack;
MainLoop;
```

هذا هو كود الراديو بوتن ونلاحظ من هذا الكود انه لايوجد اي اختلاف بينه و الزر الذي قبله الا في ناحية اسم الزر على و عند التنفيذ سوف نحصل على هذه الصورة



*FIGURE(9)

الان لدينا سؤال يطرح؟؟ وهو هل من الممكن ان يتم تصييم

(2 radio buttons or 2 checkbuttons)

الجواب على هذا السؤال ممكن اكيد وكما في هذا الكود الاتي الذي يوضح العملية

*CODE(17)

```
use Tk;
$top=MainWindow->new;
$a=$top->Radiobutton (      -text              =>"Hello world"
);
$b=$top->Radiobutton  ( -text                  =>"perl is made for wisedom seekers");
$a->pack;
$b->pack;
MainLoop;
```

الان عند تنفيذ هذا الكود سوف نشاهد الصورة التالية



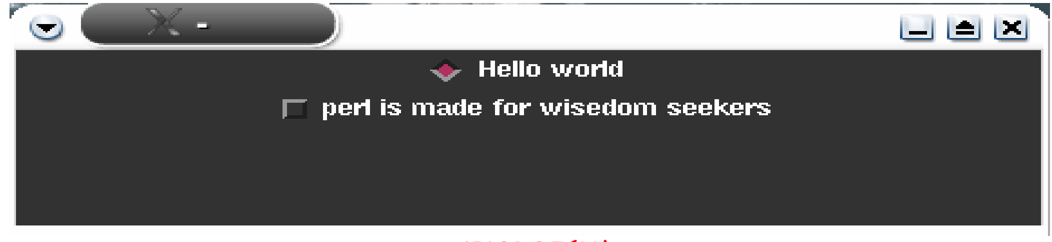
*FIGURE(10)

كما نلاحظ اذن من الممكن ان تتم عملية الدمج بين زرين من نوع واحد ولكن بعد ان تمت العملية بنجاح هذا الكود الذي كتب قبل قليل يقودنا الى فكرة اخرى هي انه نحن الان قمنا بكتابة كود هذا الكود يقوم بعمل استعراض لزرين من نوع الراديو بوتن
الان سوال اخر هو هل انه من الممكن ان يتم عمل ما يلي
(*merge between both of radio button and the checkbutton in one code*)
طبعا اكيد ومن الممكن ان تتم هذه العملية من خلال هذا الكود الذي يوضح الفكرة وكما يلي

*CODE(18)

```
use Tk;
$top=MainWindow->new;
$a=$top->Radiobutton (      -text              =>"Hello world"
);
$b=$top->Checkbutton  ( -text                  =>"perl is made for wisedom seekers");
$a->pack;
$b->pack;
MainLoop;
```

الان كما نلاحظ على ما يبدو ان العملية تمت و الان سوف نلاحظ ما هو ناتج هذه العملية لي الصورة التالية



*FIGURE(11)

اذن اصبح من الممكن ان تتم عملية دمج بين نوعين مختلفين من الازرار

الان وبعد ان اخذنا نظرة عامة عن الازرار وعن كيف نقوم بتكوينها وعن كيفية تصميم كل من النوعين المتمثلين بال

Buttons properties

الان سوف نتكلم عن خواص هذه الازرار وعن بعض التطبيقات المتقدمة التي من الممكن ان نقوم بها لكي نصبح على اطلاع اكبر بهذا الازرار وكما

الخاصية الاولى
في هذا البرنامج سوف نعمل على كتابة كود يعمل على انشاء ازرار من الراديو تقوم بعملها بشكل صحيح لانه كما لاحظت في الكودات السابقة انه الازرار بعد التنفيذ ذاتيا تظهر وقد تم تأشيرها ام الان سوف لن تظهر وقد تم تأشيرها بل أنت سوف تعمل ما تحب والكود هو كما يلي

*CODE(19)

```
use Tk;
$top=MainWindow->new;
$b=$top->Label(          -text           =>"what is ur favourite programming language"
);
$b->pack;
foreach (qw/c perl python ruby/){
$a=$top->Radiobutton(
                    -text          =>$_,
                    -variable      =>\$spawn,
                    -value         =>$_,
);

$a->pack;
}
MainLoop;
```

والان وعند تنفيذ هذا الكود سوف تظهر لنا هذه الصورة للبرنامج على كل بعد ان نرى الصورة سوف نشرح ما هو هذا البرنامج ماهي اهمية الخطوات التي قد وضعت في داخله



*FIGURE(12)

الان نلاحظ هذه الصورة ونأتي الان الى شرح البرنامج هذا البرنامج في البداية مكون من لابيل هذا ال لابيل يعمل من خلال اسناد احدى الخواص المتعلقة به ان يعمل على كتابة النص الموجود في الاعلى و هو **اختار لغتك**
البرمجية المفضلة ونلاحظ بعد ذلك انتهاء مهمة هي قمنا بالانتقال الى القسم الثاني من البرنامج و هو قسم الازرار التي هي من نوع راديو بوتون نلاحظ انه قد قمنا بوضع عبارة الفور وبداخلها اسماء اللغات ولكن ليست عبارة الفور هي ما يهمنا لانه هي عبارة عن جملة برمجية بسيطة قد مرت علينا عدة مرات و عملنا ما هو عملها ولكن ما يهمنا الاما هي المعلومات الموجودة داخل الاقواس الخاصة بالزر على كل هذه المعلومات هي فائدتها كما يلي الخيار الاول و هو كما في الكود الاتي

CODE(20)

```
-text        =>$_,
```

هذا الخيار هو الخيار الخاص بادخال اسماء لغات البرمجة الموجودة في عبارة السيطرة فور الى البرنامج ونلاحظ انه قد تم اختيار المتغير الافتراضي ال(default variable)
لانه لو قمنا باختيار متغير اخر يعني متغير نحن نحدده عدا هذا المتغير فأن الذي سوف يحدث هو انه اما اسماء اللغات لن تظهر بجانب الازرار الخاصة بها او ان جميع الاسماء سوف تظهر على زر واحد ولكن غالبا في مثل هذه الحالات فأن الخيار الاول اي عدم ظهور اسماء اللغات هو يكون الغالب لذا ارجو الانتباه الى هذه الفقرة الخيار الثاني هو التالي

CODE(21)

```
-variable    =>\$spawn,
```

هنا هذا الخيار كما نلاحظ من اسمه ان يحتوي على متغير ولكن من اين جأت قيمة المتغير؟؟؟
وان البرنامج كله لا يحتوي على القيمة المذكورة في هذا الخيار
طبعا سؤال مثل هذا السؤال هو سؤال طبيعي لانه فعلا لا توجد هكذا قيمة ولكن هذه القيمة نحن نقوم بادخالها كيف نقوم بذلك ؟؟ و هذا الخيار اي خيارالفاريابل يتم ادخاله لكي يتم التعامل مع قيمة متغير موجودة في البرنامج من دون الحاجة الى التعامل معها بشكل صريح في البرنامج يعني ترجمة كلامي هذا هو انه عندما نقوم باستعمال عبارة الفور دائما يتم استعمال متغير معها لكي تتم العملية بشكل صحيح ولكن الان لاحظ معي هذا الكود البسيط جدا

CODE(22)

```
@a=(1,2,3,4,5,6,7,8,9);
foreach $a(@a){
print $a;
}
```

لاحظ معي برنامج سهل يحتوي على مصفوفة مكونة من تسع عناصر من خلال استعمال عبارة الفور تم اسناد قيمة متغير الى المصفوفة ومن ثم قمنا بطباعة هذا المتغير
الان اتمنى ان تتوضح الصورة يعني مسبقا الان علمنا انه خيار الفاريابل هو خيار يحمل قيمة لمتغير دون الحاجة الى اسنادها بشكل صريح والان علمنا انه المتغير سباون هو مرادف للمتغير($a)
الموجود في البرنامج البسيط الموجود في الاعلى اذن المتغير سباون هو المتغير اذي تم اسناده الى اسماء اللغات الموجودة في عبارة الفور الان الصورة بدأت تنجلي وعرفنا ما الذي حصل ولكن بقيت الان فقرة واحدة غيرواضحة هي لماذا لانريد ان نقوم باسناد متغير بشكل صحيح في البرنامج ؟؟
طبعا هذا السؤال جاوبنا عنه مسبقا!!!!

لانه من قبل نحن قد قبلنا انه لن نطلق على اي ناتج تنفيذ برنامج انه طباعة ولكن سنقول عرض حيث انه لو اسندنا قيمة بشكل مباشر فانه لن تستعرض بل ستطبع ولن تكون النتيجة بصرية هذا هو السبب
اما الان ناتي الى الخيار الثالث وهو كما في الكود الاتي

*CODE(23)

```
-value          =>$_,
```

وهنا في هذا الخيار هو الخاص بالقيمة وقد تم اسنادها الى قيمة المتغير الافتراضي
الان انتهينا من عمل دمج بين خاصيتين ولكن الان سوف ندخل في خواص اخرى من خواص الازرار
الخاصية الاولى
تلوين الحدود الخاصة بالازرار الموجودة في البرنامج لدينا وتتم العملية كما في هذا الكود الاتي لدينا

*CODE(24)

```
use Tk;
$top=MainWindow->new;
$a=$top->Checkbutton(-text =>"Hello World",          -highlightbackground =>red);
$a->pack;
MainLoop;
```

نقوم بتنفيذ البرنامج ونلاحظ ما يلي عند تنفيذ البرنامج نلاحظ الصورة الاتية كما يلي



*FIGURE(13)

نرى الان انه تم احاطة الكلمة هلو ورلد بألطار احمر على طول الكلمة وهذه الخاصية ان امكن القول هي ليست خاصية برمجية ولكن هي عبارة عن خاصية تكميلية والغاية منها هي جمالية الزر الذي نعمل على برمجته

الخاصية الثانية
هذه الخاصية هي خاصية لها علاقة بالتأشير على الزر وتكون على حالتين ما 1 او صفر و الان سوف نأخذ الكود الخاص لهذه الخاصية هو الاتي

*CODE(25)

```
use Tk;
$top=MainWindow->new;
$a=$top->Checkbutton (-text                =>"Hello world",
                      -onvalue        =>0,
                      );
$a->pack;
MainLoop;
```

هذه الخاصية هي تكون تعني عندما تكون مصمم الزر فأنها ألتأشير سوف يكون لمرة اي لا يمكن ان يتم اعادة التأشير كما تريد تذكر لمرة واحدة فقط اما اذا كانت القيمة تساوي واحد فأن العملية سوف تتم

كما تريد اي تقدر ان تأشر على المربع بالعدد الذي تريده
الخاصية الثالثة
هذه الخاصية هي من الخواص البرمجية اي انها ليست من تلك الخواص التكميلية التي مر علينا ذكرها في المرات السابقة و هذه الخاصية اسمها خاصية الحالة اي حالة الزر من ناحية كونه زر فعال او زر غير فعال الى الخ طبعا الكود الخاص بهذه العملية هو الكود الاتي

***CODE(26)**

```
use Tk;
$top=MainWindow->new;
$a=$top->Checkbutton(      -text             =>"Hello world",
                           -state            =>active,
                           );
$b=$top->Checkbutton(      -text             =>"Hello world",
                           -state            =>disabled,
                           );
$a->pack;
$b->pack;
MainLoop;
```

الان نلاحظ انه عندما تنفيذ الكود الذي كتبناه في الاعلى ينتج عن التفيذ هذه الصورة ولك ان تلاحظ ماهو الفرق



***FIGURE(14)**

لاحظ الان كم هو الفرق واضح اي انه من الاكيد وكما هو ملاحظ من الصورة ان الزر الاول هو الزر الذي كانت حالته مفعلة اي اكتيف و الزر الثاني هو الزر الذي كانت حالتة غير مفعلة اي انها مبطلة

الخاصية الرابعة
هذه الخاصية من الخواص المتقدمة التكميلية ولكنها في نفس الوقت برمجية وتدل على ان الشخص الذي يستعملها هو شخص يسعى الى الكمال و الجمالية في البرنامج الذي برمجه هذه الخاصية هي خاصية تلوين السطر بلون مختلف مع كل اختيار يتوافق مع العمل الذي اقوم به و هذه تتم العملية كما يلي في هذا الكود

CODE(27)

```
use Tk;
$mw=MainWindow->new;
$rb_value = "red";
$mw->configure(-background => $rb_value);
foreach (qw/red yellow green blue grey/) {
  $mw->Radiobutton(-text => $_,
                 -value => $_,
                 -variable => \$rb_value,
                 -command => \&set_bg)->pack(-side => 'left');
}
sub set_bg {
  print "Background value is now: $rb_value\n";
  $mw->configure(-background => $rb_value);
}
MainLoop;
```

نأتي الى شرح الكود ماهي هذه الخطوات وما الذي تكون مسئولة عنه في البرنامج الخطوة هذه

اولا

CODE(28)

```
$rb_value = "red";
```

عمل هذه الخطوة هي انه لدينا متغير هذا المتغير هو متغير عادي كما نرى يحمل نص بين علامات الاقتباس للون الاحمر سنعلم معلومات اكثر عن ماهية هذه الفقرة في الخطاوت اللاحقة

ثانيا

الخطوة التي في هذا الكود

CODE(29)

```
$mw->configure(-background => $rb_value);
```

هذه الخطوة هي عبارة عن خطوة تعريفية اي انه في الخطوة السابقة ذكرنا انه لدينا متغير يحمل سترينك تحمل اللون الاحمر على انه القيمة المعطاة لها اما في هذا الخطوة كما ذكرنا انها خطوة تعريفية اي انها تم اعتماد الخلفية للبرنامج على انها اللون الاحمر اعتمادا على ما تم ادخالة للمتغير الذي اسندت له قيمة اللون الاحمر اي الان الخلفية الخاصة بالبرنامج هي احمر ولكن مما تجدر الاشارة اليه هي انه لون تم تغيير اللون الاحمر في اول خطوة قمنا بشرحه مثلا الى اللون الازرق فأن الخلفية الخاصة للتفيذ البرنامج سوف تتحول من اللون الاحمر الذي تم اعتماده مسبقا الى اللون الازرق وهكذا العملية

ثالثا

CODE(30)

```
foreach (qw/red yellow green blue grey/)
```

هذه الخطوة لاتحتاج الى شرح مجرد جملة فور تحتوي بداخلها اسماء الألوان التي سنعمل عليها وفقط لا اكثر و لااقل

رابعا
شرح هذه الخطوة هو

CODE(31)

```
$mw->Radiobutton(-text => $_,
        -value => $_,
        -variable => \$rb_value,
        -command => \&set_bg)->pack(-side => 'left')
```

الان هذه الخطوة كما بات معروفا لدينا هو انه هي مسئولة عن تكوين الازرار و التي في حالتنا هذه سوف تكون من نوع الراديو بوتن اما
المعلومات التي تحتويها هي خيار ال تيكست وهو تحميل اسماء الالوان الى الازرار وتم استعمال المتغير الافتراضي في هذه الحالة لانه كي لايتم حصول خلل في ترتيب الاسماء الخاصة بالالوان
اما خيار الفاريابل فهو الخيار الخاص الذي ذكرنا ما هو عمله في بداية الشرح عن دمج خواص ال لايبل مع خواص الازرار اما الكوماند ماهو ماعمله ايضا شرحنا ماهو عمله في بداية شرحنا على عمل الازرار
خامسا
اما الان فأن البرنامج الرئيسي او ما يحب ان يسميه المبرمجون الاخرون البرنامج الاول قد انتهى وحان الوقت للعمل على البرنامج الثانوي او الثاني او الاصح برمجيا حان الوقت للعمل على الروتين الفرعي وهو كما يلي في الكود الاتي

CODE(32)

```
sub set_bg {
 print "Background value is now: $rb_value\n";
 $mw->configure(-background => $rb_value);
}
```

عمل الروتين الفرعي هو كما يلي في البداية لدينا رويتن فرعي متكون من خطوتين واسمه اي اسم الروتين الفرعي هو(set_bg)
وقد تم ادخاله الى الى مكونات الزر البرمجية من خلال الامر او الخيار كوماند ومن ثم
اول شئ يحتويه هوجملة الطباعة وهي تحتوي على نص بسيط وهذا بسيط ومفهوم وهي ايضا في نفس الوقت تحتوي على المتغير الذي قد قمنا بادخاله في بداية البرنامج والغرض منه هو انه عند التحويل من لون الى لون فأن جملة الطباعة تخبرك انه تم التغير الان من اللون الاحمر الى اللون الازرق على سبيل المثال وهكذا دواليك
اما الخطوة الاخرى من الروتين الفرعي هي الخطوة التي تؤكد على استمرار تعريف الخلفية الخاصة بالبرنامج
اما الان حان وقت تنفيذ البرنامج لكي نرى ما هو ناتج التنفيذ من هذا البرنامج
واليكم الصور الخاصة بهذه العملية



*FIGURE(15)

طبعا اضطريت الى ان اقوم بتفيذ البرنامج اكثر من مرة كما هو ملاحظ لكي تقوموا بملاحظة الفرق على كل انتهينا من هذه الخاصية من ناحية الصورية الان حان الوقت لكي نرى الفرق من خلال الشيل كي نرى كيف يتم التبديل بين الالوان والصورة التالية سوف توصف هذا

```
Background value is now: yellow
Background value is now: green
Background value is now: blue
Background value is now: grey
Background value is now: red
Background value is now: yellow
```

*FIGURE(16)

هنالك ملاحظة احب ان انوه اليها في البرنامج هي ان الخطوة هذه المدرجة في الكود

*CODE(33)

```
$mw->configure(-background => $rb_value);
```

ومن ان الممكن ان يتم الاستغناء عنها لانه هذه الخطوة مكررة في الروتين الفرعي لذا من الممكن ان تقوم بالاستغناء عنها

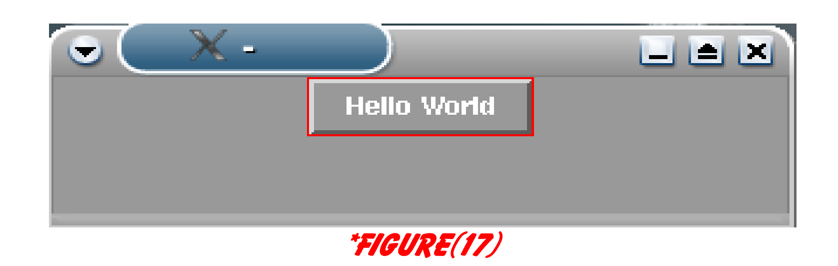
الخاصية الرابعة

قبل ان ندخل في اعماق هذه الخاصية سوف نتعلم تقنية جديدة سوف تكون مفيدة في هذه الخاصية وهذه التقنية هي تقنية الزر المفرد اي انشاء زر ليس من نوع التشيك بوتن وليس من نوع الراديو اي زر مفرد لوحدة والكود الخاص بأنشاء هكذا نوع من الازرار هو كما يلي

*CODE(34)

```
use Tk;
$top=MainWindow->new;
$a=$top->Button(     -text                          =>"Hello World",
                     -highlightbackground =>red,
               );
$a->pack;
MainLoop;
```

عند تنفيذ الكود سوف نحصل على نتيجة كما هي مدرجة في الصورة ادناه



*FIGURE(17)

نلاحظ ان هذه الصورة التي هي ناتج تنفيذ البرنامج السابق لاتحتوي على اي تعقيد و هي ليست بمقدار من الصعوبة
الان بعد ان تعملنا تقنية الزر المفرد هذه فأنه سوف نرجع الى الخاصية التي سبق ان تكلمنا عنها وهي خاصية التحكم و التلاعب بمحتويات الزر الذي نعمل على انشأه وذلك يتم كما في الكود المدرج ادناه

*CODE(35)

```
use Tk;
$top=MainWindow->new;
$value = 0;
$cb = $top->Checkbutton(-text => "Checkbutton",
                -variable => \$value,
                -command => sub { print "Clicked! $value\n" }
                )->pack(-side => 'top');

$top->Button(-text => "Value on",
        -command => sub { $value = 1 })->pack;
$top->Button(-text => "Value off",
        -command => sub { $value = 0 })->pack;
MainLoop;
```

الان نأتي الى شرح هذا الكود لكي نفهم ماهي مكونات الخطوات ومالذي ينتج منه وكما يلي
الخطوة الاولى

*CODE(36)

```
$value = 0;
```

لن اعيد في هذه الخطوة كثير مجرد متغير عادي يحمل القيمة
الخطوة الثانية
شرحها في هذا الكود الاتي

*CODE(37)

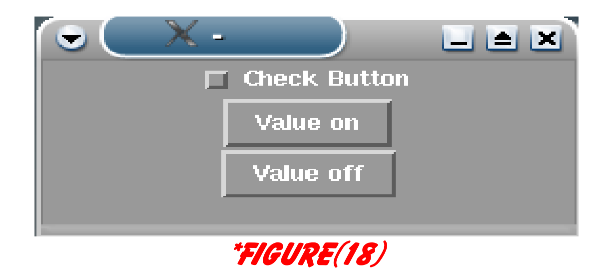
```
$cb = $top->Checkbutton(-text => "Checkbutton",
                -variable => \$value,
                -command => sub { print "Clicked! $value\n" }
                )->pack(-side => 'top');
```

الان شرح المعلومات داخل البلوك الزر
التكست مفهوم وشرحناه اكثر من مرة
اسناد قيمة الفاريبل الى متغير موجود في البرنامج من اجل العمل على قيمة ال
(on and the off value in the code)
الخطوة الثالثة وهي الكوماند وهي كما نلاحظ انها تحتوي على روتين فرعي
على كل عندي ملاحظة قل ما نشرح هذه الخطوة وهي انه كل مابين قوسين ويكون هذين القوسين من
نوع ال(Curly Brackets)
اي من نوع ال{}
وتكون مسبوقة بكلمة(sub)
هذا يعني ان البوك الذي امامنا هو عبارة عن روتين فرعي دائما وابدا
نرجع الان الى نقطتنا الاساسية المهم نلاحظ دائما ايضا انه الامر كوماند يأتي معه الروتين الفرعي وهنا في
البرنامج الذي كتبناه
وهنا وظيفة الكوماند هي انه يطبع كلمة(printed)
وايضا قيمة المتغير($value)

*CODE(38)

```
$top->Button(-text => "Value on",
        -command => sub { $value = 1 })->pack;
```

اما الان شرح هذا الجزء من الكود هو انه يتم انشاء زر مفرد توجد عليه كلمة(value on)
ومن ثم الامر كوماند هذا الامر قد تم اسنادها الى روتين فرعي ويحمل في داخل هذا الروتين قيمة المتغير
المفترض في بداية البرنامج ولكن تم تفعليها الى واحد
اما الخطو التي تأتي بعد هذه الخطوة هي مثل هذه الخطوة ولكن تختلف عنها هي انه الخطوة الاخرى قيمة المتغير
فيها هو صفر وفقط
والان عند تنفيذ هذا البرنامج نحصل على الصورة الاتية



*FIGURE(18)

بعد ان انتهينا من كل ما يتعلق بال(*Buttons*)
في لغة البيرل سوف ننتقل الى تقنية جديدة

EntrY

الان بعد ان دخلنا في برمجة الواجهات الرسومية لاحظت انه لم نأخذ من وسائل الادخال سوى وسيلتان هما
1-Labels
2-Buttons(Buttons,Checkbutton,Radiobutton)
وفقط الان سوف نأخذ طريقة او تقنية اخرى و هي تقنية تشبه تقنية ال(*Labels*)
على كل سوف نأخذ الان كود بسيط يوضح عمل تقنية ال(*Entry*)
والكود الاتي سوف يشرح عمله

CODE(39)

```
use Tk;
$top=MainWindow->new;
$a=$top->Entry(            -text                    =>"hello world",
                );
$a->pack;
MainLoop;
```

وقبل ان اعرض الصورة هذا البرنامج عبارة عن برنامج بسيط يعني لايحتوي اي شئ من التعقيد الذي يحتاج الى شرح على كل هذه صورة البرنامج



FIGURE(19)

ولكن عند تنفيذ البرنامج من احد أهم الفروق التي نلاحظها على البرنامج و التي لم نلاحظها في البرامج والتقنيات السابقة هي انه من الممكن ان يتم تغيير النص المكتوب داخل ال(*entry*)
حتى بعد ان يتم تنفيذ البرنامج يعني الان من الممكن ان تغير كلمة(*hello world*)
الى اي كلمة اخرى مثلا تغيرها الى اسمك او الى اخره من الخيارات
الان بعد ان انتهينا من عمل التعريف العام الخاص بتقنية ال(*Entry*)
الان سوف ندخل الى برمجة الخواص الخاصة بال(*Entry*)

الخاصية الاولى
طبعا في بداية الكلام عن الخصائص فأنا لن اتكلم عن جميع الخصائص لأن اغلبها الان اصبح مكرر وبالأضافة الى هذا ان اغلب هذه الخصائص اصبحت مشتركة بين كافة الوسائل التي اخذناها
على كل ندخل الان في خاصية
هذه الخاصية هي خاصية ادخال لون وهي تكون كما يلي في هذا الكود

*CODE(40)

```
use Tk;
$top=MainWindow->new;
$a=$top->Entry(             -text                    =>"Hello world",
                       -insertbackground      =>red,
               );
$a->pack;
MainLoop;
```

الان عند تنفيذ هذاالكود سوف نحصل على هذه الصورة الخاصة بتنفيذ البرنامج



*FIGURE(20)

اذن هذه الخاصية هي خاصية تعتبر ذات اهمية اذا اردت ان تغير لون المؤشر في داخل ال(entry)
ولكن مما تجدر الااشارة اليه هي انه لا يتم تغير لون الاحرف عند الكتابة بالمؤشر الملون اي ان الحرف سوف يبقى يكتب باللون الذي يكتب به ولن يتغير لونه الى اللون الاحمر

اكيد ومن الممكن ان يتم استعمال خاصية الباك كروند وخاصية الفور كروند مع هذه الخاصية لكن لن اشرحها لكي لا يطول الشرح في خواص قد سبق و ان شرحناها

الخاصية الثانية
هذه الخاصية هي خاصية الحالة اي الحالة التي يكون فيها حالة ال(entry)
مفعلة او ليست مفعلة
الشرح في هذا الكود

*CODE(41)

```
use Tk;
$top=MainWindow->new;
$a=$top->Entry(          -text                    =>"Hello world",
                           -state                   =>disabled,
);
$a->pack;
MainLoop;
```

الان نلاحظ انه عند تنفيذ البرنامج انه عندما نأتي الى ما عرفناه من تغير النص داخل صندوق ال
(Entry)
سوف لن نكون قادرين على هذا التغيير والسبب يعود في هذا الوضع هو انه قد عطلنا خاصية التغيير
هذه من تقنية ال(Entry)
والصورة الخاصة بتنفيذ البرنامج هي كما يلي



*FIGURE(21)

من الوهلة الاولى عند النظر الى البرنامج نلاحظ الاختلاف بينه وبين البرنامج الاخر الذي سبق تنفيذه من ناحية
الصورة بالأضافة الى الفرق البرمجي بين البرنامجين

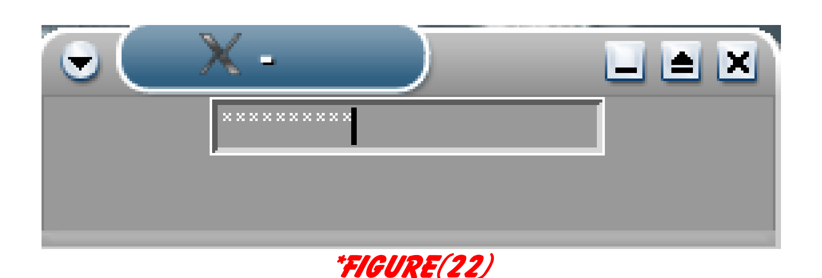
الخاصية الثانية
هي خاصية ال(relief)
او ما يعرف بخاصية الزخرفة ومن الممكن ان يتم تطبيق كافة خياراتها على تقنية ال(entry)
ولكن لن اقوم بشرحها لانني قد شرحتها في تقنية ال(labels)
ولكي نتجنب ال الشرح الطويل و المكرر

الخاصية الثالثة
وهذه الخاصية هي من اهم الخصائص الموجودة في تقنية ال(*entry*)
وهي خاصية التشفير ويكون عمل هذه الخاصية من خلال خيار العرض ويتم استعمال هذه الخاصية كما يلي في هذا البرنامج

CODE(42)

```
use Tk;
$top=MainWindow->new;
$a=$top->Entry(-show                    =>"*",
               );
$a->pack;
MainLoop;
```

هذا البرنامج يعمل على تشفير الاحرف التي نعمل على ادخالها ويعمل على تشكيلها على شكل نجوم ولكن الاشارة اليه هي انه نحن نجدد المعيار الذي يتم على اساسه التشفيراي لو تم تغيير الحرف او المعيار ذو شكل النجمة الى شكل اخر اي الى حرف اكس مثلا فأن الاحرف و الكلمات التي سوف ندخلها سوف تأخذ شكل حرف الاكس على كل شكل هذا البرنامج بعد التنفيذ هو كما يلي



FIGURE(22)

الخاصية الرابعة
هذه الخاصية هي خاصية خاصة بالالوان ولن اتكلم عنها لانها قد شرحت من قبل كل من هذه الصفات ومشتقاتها من الممكن ان يتم تطبيقها على تقنية ال(*entry*)
وعلى كل هذه الخصائص هي

1-highlightbackground
2-highlightcolor
3-insertbackground

الان انتهينا من خاصية التعريف بالالوان وتحديد الخلفيات الخاصة بال(*entry*)

الخاصية الخامسة

هذه الخاصية هي من الخواص التي تكون مسئولة عن تحديد البعد الذي يحيط بصندوق ال(entry)
ولكي تصبح الفكرة مفهومة اكثر اليكم الكود الخاص بهذه العملية

*CODE(43)

```
use Tk;
$top=MainWindow->new;
$a=$top->Entry(          -text                    =>"Hello world",
                         -highlightthickness =>15,
               );
$a->pack;
MainLoop;
```

الان وبعد انيتم تنفيذ الكود نلاحظ انه سوف نحصل على الصورة الاتية من بعد التنفيذ
واليكم الصورة



*FIGURE(23)

ما تجدر الاشارة اليه في هذه الخاصية هي انه كل مرة كبرت فيها الرقم الموجود مع هذه الخاصية كل
ما زادت الهالة البيضاء التي تحيط بال(entry)
اي انه العلاقة بينهما علاقة طردية

الخاصية السادسة
هذه من اكثر الخواص التي قد تراها غرابة وهي على العموم غريبة وطريفة في نفس الوقت عمل
هذه الخاصية هي تحدد ان امكن القول تردد المؤشر اذا ما ادخل الى صندوق ال(*entry*)
وهذا طبعا يكون اعتمادا على الزمن الذي تدخله والمعيار الزمني الذي يقاس عليه هو زمن ال
(*Milliseconds*)
الكود الذي يشرح هذه العملية هو الكود الموجود ادناه

CODE(44)

```
use Tk;
$top=MainWindow->new;
$a=$top->Entry      (            -text              =>"Hello World",
                                   -insertontime       =>10,
                  );
$a->pack;
MainLoop;
```

الان نأتي الى شرح الكود هذا هو كود على العموم سهل وبسيط ولكن اهمية الخيار(*insertontime*)
هي انه عندما نقوم بألتأشير على صندوق ال(*entry*)
بؤشر الفأرة فأنه سوف يصبح من الممكن لنا ان نحرر في مكونات النص الموجودة داخل صندوق الانتري
الان هنا هذا الخيار اهميته هو تجعل المؤشر الخاص بالفأرة ينبض
(*Every 10 milliseconds*)
طبعا على العموم هذه الخاصية لن يكون معها صورة مرفقة لانه الفرق الذي سوف تحصل عليه لن يكون من الممكن ان تلاحظه من خلال صورة جامدة
ولكن اذا اردت ان تلاحظ الفرق بصورة جيد وعملية هي تكمن في كتابة كودين اول من دون هذه الخاصية و الكود الاخر يكون مع هذه الخاصية لكي تلاحظ الفرق من المرة الاولى وببساطة

الخاصية السابعة
هذه الخاصية التي سوف نتناولها الان هي ايضا خاصية تتعلق بالالوان لكنني لم ادرجها في الخاصية الخاصة بألوان الموجودة في الصفحات القليلة السابقة لانه هذا النوع من الخواص لم نكن قد تطرقنا اليه مسبقا على كل الخاصية اسمها(selectbackground)
و عملها سوف تلاحظه من خلال هذا الكود

*CODE(45)

```
use Tk;
$top=MainWindow->new;
$a=$top->Entry(          -text                    =>"Hello world",
                           -selectbackground          =>red,
                  );
$a->pack;
MainLoop;
```

قبل ان نقوم بتنفيذ الكود الخاص بهذه الخاصية هنالك خاصية اخرى تندرج تحت هذه العائلة اي عائلة التحكم بالوان و هي خاصية(selectforeground)
والكود الخاص بها هو الكود الاتي الذي يوضح ماهو عملها

*CODE(46)

```
use Tk;
$top=MainWindow->new;
$a=$top->Entry(          -text               =>"Hello world",
                  -selectforeground      =>yellow,
);
$a->pack;
MainLoop;
```

الان بعد ان قمنا بتنفيذ هذين الكودين سوف نعرض الصور الخاصة بتنفيذهما
الان الصورة الخاصة بخاصية ال(Selectbackground)



*FIGURE(24)

اما الان فالصورة الثانية الخاصة بخاصية ال(selectforeground)



*FIGURE(25)

لكن دون الحاجة الى شرح كل من الخاصيتين من الممكن ملاحظة ماهو عملها ومالذي تعمله على صندوق ال(*entry*)
من خلال المشاهدة فقط

Scroll Bars

بعد ان اتهينا من الكلام عن خاصية وتقنية ال(entry)
سوف ننقل رحالنا الى كل ما يتعلق بكل خواص السكرول بار وما يتعلق بها وما تقوم به وكيفية برمجتها واهم خواصها الخ من التقنيات و الخواص المتعلقة بها
كل من يستخدم الكمبيوتر يعرف ماهو السكرول بار وكلنا نعرف انه عبارة عن خط مستقيم او عمود اشبه بالمسطرة يوجد على جانب الصفحة يعمل على تحريك الصفحة الى الاعلى او الى الاسفل وهذا النوع من السكرول بار يطلق عليه اسم(vertical Scrollbar)
اما النوع الثاني من السكرول بار هو النوع الذي يوجد في الاسفل الصفحة الذي يعمل على تحريك الصفحة الى اليمين و اليسار وهذا النوع من السكرول بار يطلق عليه اسم ال(horizontal Scrollbar)
الان سوف نكتب برنامج بسيط يوضح كيف هو شكل السكرول بار وكيف يعمل اليكم الكود

CODE(47)

```
use Tk;
$top=MainWindow->new;
$a=$top->Scrollbar(-orient  =>'vertical',
                   -width  =>30,
);
$a->pack;
MainLoop;
```

نظرة سريعة على البرنامج هو عبارة عن برنامج بسيط يحتوي في البداية على المعرف سكرول بار الذي نحن نريد تكوينه ومن ثم داخل البلوك البرمجي لدينا خيار الاورينت وهذا الخيار من خلاله تحدد نوع السكرول بار الذي تريده ان يكون موجود في برنامجك اي ان يكون السكرول بار من النوع الاول اي سكرول بارعمودي او يكون من النوع الثاني اي سكرول بار أفقي اما الخيار الثاني فهو من
خلال هذا الخيار تحدد العرض الذي تريده ان يكون للسكرول بار الذي برمجته ان يكون عليه وكل مرة زاد مقدار الرقم فيها كلما زاد حجم السكرول بار اليكم الصورة الخاصة بتنفيذ البرنامج



FIGURE(26)

اما اذا اردت ان تكون السكرول الخاصة بالبرنامج الذي تبرمجه ان تكون افقية فقط غير ال اتجاه خيار الاورينت من
(Vertical to horizontal and that`s it dude)
الان سوف نأخذ الخاصيات الخاصة و المتعلقة بتقنية السكورل بار

الخاصية الاولى
هنا في هذه التقنية اغلب الخاصيات التي شرحناها من الممكن ان يتم تطبيقها على هذه التقنية مثل

1-relief
2-highlightcolor
3-highlihgtthickness
4-orient
5-width
6-background

ولكن هذه ليست كل الخاصيات المتعلقة بهذه التقنية هنالك بعض الخواص المتقدمة للهذة التقنية سوف يتم شرحها لاحقا وعلى اثرها سوف نبرمج كل ما يتعلق في السكرول بار

Advanced application on Scroll bar Technic

سوف نبرمج عدد من البرامج المتفرقة الخاصة ببرمجة السكرول بار
البرنامج الاول
الكود الخاص بهذا البرنامج هو الكود الاتي

*CODE(48)

```
use Tk;
$top=MainWindow->new;
$a= $top->Scrollbar(-orient => 'horizontal');
$b= $top->Entry(-text          =>"my web site is programming-fr34ks.net,see me there",
                    -width => 30,
               -xscrollcommand => ['set' , $a]);
$a->configure(-command => ['xview', $b]);
$a->pack(-side => 'bottom', -fill => 'x');
$b->pack(-side => 'bottom', -fill => 'x');
MainLoop;
```

ألأن نأتي الى شرح هذا الكود لكي نعرف ما الفائدة منه
الخطوة الاولى

*CODE(49)

```
$a= $top->Scrollbar(-orient => 'horizontal');
```

هذه الخطوة هي خطوة بسيطة عملها ان تقوم بأنشاء سكرول بار من النوع الافقي فقط
الخطوة الثانية

*CODE(50)

```
$b= $top->Entry(-text          =>"my web site is programming-fr34ks.net,see me there",
                    -width => 30,
               -xscrollcommand => ['set' , $a]);
```

اما عن هذه الخطوة فهي مسئولة عن تكوين صندوق ال(entry)
الذي شرحناه في الموضوع السابق حيث نلاحظ انه يحتوي على المعلومات الاتية(text)
الخيار الذي يعمل على ادخال النص الذي نريده الى صندوق ال(entry)
الخيار الثاني هو خيار تحديد عرض ال(entry box)
وكلما زاد الرقم الخاص بهذا الخيار كلما زاد عرض صندوق ال(entry)
اما الخيار الثالث
فهو الخيار الخاص الذي يتم اسناد عمليات ال(callback)
اليه لكي يتم العمل على السكرول بار والقدرة على تحريكها اذن ان امكن القول هو الخيار الخاص الذي يعمل على اعطاء السكرول بار الذي عرفناه في الاعلى القدرة على الحركة الى اليمين واليسار
ولكن لو تلاحظ بدقة ان السكرول بار الناتجة من هذا البرنامج لن تفقد القدرة على الحركة بشكل تام مئة في المئة ولكن سوف يكون لها القدرة على التحرك الى جهة اليمين مقدار حرف واحد فقط لا غير ولذلك جرب ان تنفذ البرنامج بعد ان تلغي هذه الخطوة ولك ان تلاحظ ما هو الفرق
وسوف يكون لها القدرة على الرجوع ايضا مسافة حرف واحد الى اليسار
الخطوة الثالثة

*CODE(51)

```
$a->configure(-command => ['xview', $b]);
```

اما هذه الخطوة فهي الخطوة التي سوف تدعي السكرول بار الحركة الكلية اي انه لو الغيت سوف لن تتحرك السكرول بار نهائيا وابدا
اما الخطوات التالية فهي خطوات الخيار(pack)
وهذه الخطوات هي الخطوات الخاصة باعطاء الاتجاه و اعطاء شكل الحزك طيف سيكون ولكن على العموم هذه الخطوات مهمة ولكن من الممكن ان يتم الاستغناء عن الخيارات الموجودة مع الامر (pack)
ولكن من الافضل وجودها لانها خيارات تكميلية وتعمل على جعل البرنامج يبدو بصورة افضل
والان بعد ان عرفنا هذا الكود ماهو وماهي وظيفته اصبح لا بد ان نرى ماهي الصورة الناتجة من تنفيذ هذا الكود وهي الصورة التالية



*FIGURE(27)

الان نأخذ مزيد من البرامج الخاصة بتقينة السكرول البار لانه سوف ننتقل الى تقنية ال(Listbox)
وهذه لتقنية تحتوي على الكثير من الامور المتعلقة بالسكرول بار لذا سوف ننتقل اليها

ListBoX

تعريف عام ماهو ال(*list box*)

هو عبارة كما هو واضح من اسمها انها صندوق تحتوي على عدد من السلاسل النصية التي حيث يتم عرض سلسلة نصية لكل سطر

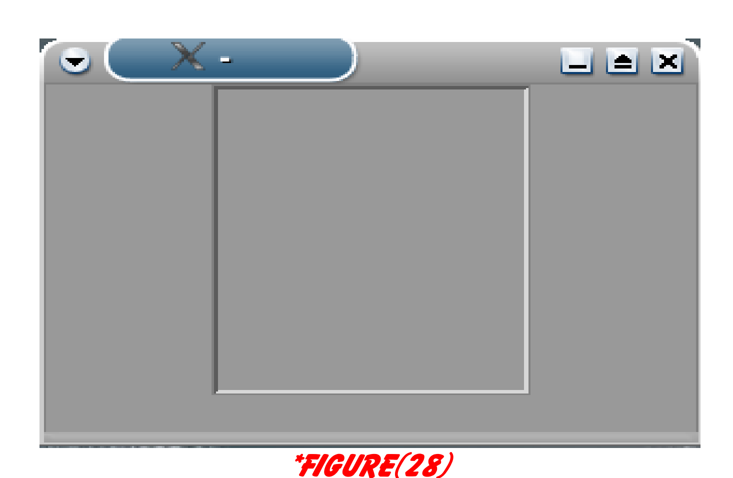
ولك الحرية ان تضع في هذا الصندوق ما تريد اي ارقام او كلمات او حروف مفردة او رموز الى اخره اي من الممكن ان تضع فيها ما تشاء وطبعا من الممكن ان تختار من هذه الرموز او الاحرف او لكي تطبق عليها ما تريد وكما سوف نلاحظ في الصفحات القليلة القادمة

الكود العام لهذه التقنية هو الكود الاتي

CODE(52)

```
use Tk;
$top=MainWindow->new;
$a=$top->Listbox();
$a->pack
MainLoop;
```

نلاحظ الان انه عندما يتم تنفيذ البرنامج سوف نحصل على الصورة الاتية



FIGURE(28)

الخواص العامة لتقنية ال(*listbox*)

هذا النوع من تقنية ال(*listbox*)

من الممكن ان يتم استعمال التقنيات السابقة مثل هذه

1-background
2-highlightcolor
3-highlightbackground
4-selectbackground
5-relief
6-font
7-foreground
8-height
9-width

كل هذه التقنيات من الممكن ان يتم استعمالها مع هذه التقنية ولكن لن اشرحها لانه قد سبق وان تم شرحها من قبل

Listbox properties

شئ غريب في الشرح عن الخواص التي تكلمنا عنها في الاسطر القليلة الماضية وهي انه لم نأتي على على شرح كيف يتم اضافة النصوص الى ال(listbox)
واضافة الى هذا ان اسلوب اضافة النص لا يتم من خلال استعمال الامر(text)
ما تعملنا ولكن يتم من خلال استخدام امر اخر وهو امر الادخال وتتم عملية اضافة النص الى تقنية ال (list box)
كما يلي في هذا الكود الاتي

CODE(53)

```
use Tk;
$top=MainWindow->new;
$a=$top->Listbox(-width              =>30);
$a->insert('end',
"www.programming-fr34ks.net",
"www.securitygurus.net",
);
$a->pack;
MainLoop;
```

الان نأتي الى شرح هذا الكود في الفقرة الاولى من وهي الفقرة الاتية في الكود الاتية

CODE(54)

```
$a=$top->Listbox(-width              =>30);
```

هذه الخطوة التي تكون مسئولة عن تكوين ال(listbox)
وتم فيها فقط تحديد عرض ال(listbox)
لكي يتسع للمعلومات التي نرغب في ادخالها واضا نلاحظ انه لن يتم تحديد اي اشارة من خلالها لكي يتم ادخال نص الى ال(listbox)
على كل ننتقل الى الخطوة الاتية وهي خطوة مهمة وهي اول مرة تمر علينا هكذا خطوة وهي الخطوة الاتية في هذا الكود

CODE(55)

```
$a->insert('end',
"www.programming-fr34ks.net",
"www.securitygurus.net",
);
```

هذه الخطوة التي من خلالها يتم ادخال نص الى صندوق العرض ومن المهم ان نعرف ما يأتي وهو ان لماذا عند تنفيذ البرنامج لا يتم عرض كلمة نهاية(end)
في صندوق العرض أيوجد خلل في البرنامج الذي كتبناه؟؟

اولا كلا يوجد خلل
ثانيا كلمة نهاية هنا هي ليست كلمة نصية وليست متغير مؤقت بل خيار ملازم دائما وابدا الى خيار(*insert*)
ودائما ضع هذه الملاحظة في تفكيرك هي انه الخيار الوحيد الذي يتم من خلاله ادخال نص الى صندوق العرض هو من خلال الامر(*insert*)
ودائما هذا الامر تكون كلمة(*end*)
مرافقه لهذا الخيار الخاص بالادخال لذا ارجو الانتباه وبعد هذا الشرح على البرنامج لم يتبقى لنا ألا ان نرى صورة البرنامج وهي كما في الشكل الاتي



*FIGURE(29)

Select mode

بعد ان تكلمنا عن اسلوب اضافة النص الى صندوق العرض نلاحظ
ظهور سطرين في صندوق العرض السطر الاول هو سطر الذي يحتوي على
www.programming-fr34ks.net
www.securitygurus.net
سوف نتكلم الان عن الخواص التي تكون ملحقة بهذه الاسطر في داخل صندوق العرض اي بعبارة اخرى كيف نقوم بالتحكم بالنصوص من داخل معرف صندوق العرض وخارج امر ال(insert)
المسئول عن امر الادخال الى صندوق العرض
هذا الكلام من الممكن ان يتم ترجمته الى لغة البيرل من خلال تقنية ملحقة بصندوق العرض تدعي بتقنية نمط الاختيار
ولهذه التقنية خيارات ملحقة بها لكي تزودك باقصى مقدار من التحكم الذي تصبو اليه والخيارات التي تملكها هذه التقنية هي

1-browse
2-extended
3-multiple
4-single

الان سوف نأتي الى شرح هذه الخواص و ما هو عملها وكيف يتم الحاقها لكي نرى ماهي الوظيفة التي تقوم بعملها وكيف نقوم بالاستفادة منها اليكم الكود الخاص بالخيار(*single*)

CODE(56)

```
use Tk;
$top=MainWindow->new;
$a=$top->Listbox(-width              =>30,
                 -selectmode    =>'single');
$a->insert('end',
"www.programming-fr34ks.net",
"www.securitygurus.net",
);
$a->pack;
MainLoop;
```

على كل هذا الكود لايحتوي على شئ غريب نأتي الى التنفيذ ورنى الصورة الناتجة منه عسى ولعله نعرف ماهي الفائدة من استعمال هذه الخاصية



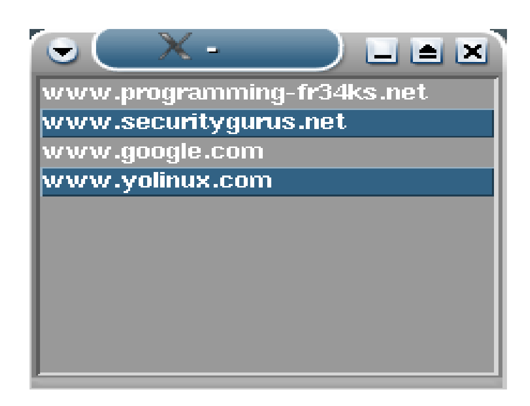
FIGURE(30)

الان ايضا من خلال النظر الى الصورة لم نلاحظ اي فرق عن الصورة السابقة الناتجة من البرنامج الماضي على سوف نترك هذا الخيار وسوف ننتقل الى الخيار الاخر وهو الخيار الذي يحمل الاسم(multiple) وسنرى عمله من خلال هذا الكود

CODE(57)

```
use Tk;
$top=MainWindow->new;
$a=$top->Listbox(-width              =>30,
                 -selectmode    =>'multiple');
$a->insert('end',
"www.programming-fr34ks.net",
"www.securitygurus.net",
"www.google.com",
"www.yolinux.com",
);
$a->pack;
MainLoop;
```

الان ايضا نأخذ ظرة عامة عن هذا الكود نلاحظ ايضا انه لايوجد فرق ولكن سنرى الناتج الظاهر من تنفيذ البرنامج لانه التغيير في هذه الحالة سوف يكون تغيير صوري و اليكم الصورة الخاصة بهذا البرنامج



*FIGURE(31)

اذن الان اخير عملنا ما هو الفرق بين هذين الحالتين والفرق اذا لم تلاحظه هو انه في الحالة الماضية حتى لو كان لديك اكثر من اختيار في صندوق العرض فأن لغة البيرل سوف لن تسمح لك بأن تختار اكثر من اختيار مهما حدث ومهما فعلت ولكن في نفس الوقت وفرت لك هذا الخيار الذي يمسح لك بأن تختار اكثر من اختيار في نفس الوقت من دون ان يعرضك او ان تواجهك اي مشاكل

اما الخيار الثالث
فهو الخيار الذي يحمل الاسم(extended)
والكود الخاص به هو الكود الاتي

*CODE(58)

```
use Tk;
$top=MainWindow->new;
$a=$top->Listbox(-width          =>30,
            -selectmode    =>'extended');
$a->insert('end',
"www.programming-fr34ks.net",
"www.securitygurus.net",
"www.google.com",
"www.yolinux.com",
);
$a->pack;
MainLoop;
```

هذا الخيار هو خليط من النوع الاول والنوع الثاني!!!
اي انه يقبل ان تختار اكثر من اختيار واحد وهو في هذه الخاصية بشبة الخيار الثاني
وهو في نفس الوقت لا يسمح لك ان تختار اكثر من خيار واحد
اذن كيف هذا يحدث؟؟
العملية تتم كما يلي اذا قمت بالنقر على كل عنصر بصورة مستقلة فأنه سوف يقوم بأختيار كل عنصر واحد لكل عملية نقر ولكن اذا عملت(hold on)
لزر الماوس وحركت المؤشر الخاص بالماوس الى الاعلى و الى الاسفل سوف تحصل على تأشير لأكثر من اختيار

ولن اقوم بادراج الصورة الخاصة به لانه لن تكون مفيدة
الخيار الرابع
فأنه من غير المفيد ان نتكلم عنه في الوقت الحالي لانه لن نحتاج اليه في الوقت الحاضر كل ما عليك ان تعرف مايلي
خيار ال(browse is very suitable with the bind operator)

Scrolling in other way

تعرفنا من خلال هذه التقنية ومن التقنية السابقة انه اذ اردت ان تقوم بأنشاء سكرول بار فأنه من الممكن ان تقوم بذلك من خلال استخدام التقنية المسماة بتقنية السكرول بار ولكن ما يحصل اذا لم تكن من محبي هذه التقنية فأن لغة البيرل توفر لك بديل اخر يمكنك من القياك بعمليات الدرجة تؤدي نفس عمل تقنية السكرول بار ولكن باستخدام تقنية اخرى سوف نلاحظ عملها من خلال هذا الكود

CODE(59)

```
use Tk;
$top=MainWindow->new;
$a=$top->Scrolled("Listbox",-scrollbars           =>'e',
                       -selectmode    =>"single",
);
$a->insert('end',
"a",
"b",
"c",
"d",
"e",
"f",
"g",
"h",
"i",
"j",
"k",
"l",
);

$a->pack;
MainLoop;
```

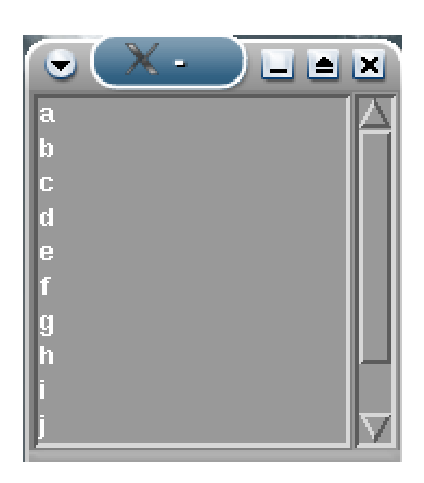
نأتي الى شرح هذا الكود وشرح اول فقرة

CODE(60)

```
$a=$top->Scrolled("Listbox",-scrollbars           =>'e',
                       -selectmode    =>"single",
);
```

اذن لاحظنا من هذا البرنامج ان التقنية البديلة تدعى ب(Scrolled)
والبيانات التي تحتويها في داخل البلوك البرمجي هي كألتالي

اولا معلومة مفردة لل(*listbox*)
حيث من خلال هذه المعلومة المفدرة الخاصة بصندوق العرض فأن مترجم البيرل سوف يعلم انه سيتعامل مع صندوق العرض اما المعلومة الثانية هي الخاصة بالسكرول بارز وهذه الملعومة هي تحدد الاتجاه الخاص بالسكرول بار وسوف نناقش الخيارات الخاصة بهذا الامر بعد قليل ثم المعلومة او الامر الاخير وهو نمط الاختيار وقد اخترناه على النمط المفرد وهذه المعلومة و الاوامر الملحقة بها قد تم شرحها بالتفصيل
ومن ثم نأتي الى الامر الخاص بالادخال وهو الامر(*insert*)
وهذا الامر سوف يتولى عملية ادخال النص الى صندوق العرض
والان حينما نقوم بتنفيذ البرنامج سوف نحصل على الصورة الاتية



*FIGURE(32)

الان هذه الصورة الخاصة بتنفيذ البرنامج وعن نفسي انا افضل هذا النوع من السكرول بار اي انا من ناحيتي افضل هذه الطريقة

How to delete

الان عرفنا كيف يتم اضافة نص الى الصندوق العرض وذلك كما ذكرنا العملية تتم من خلال استعمال الامر(*insert*)
ولكن بعد ان قمت بأدخال النص الى صندوق العرض لاحظت انه لديك خلل في احدى المعطيات او انه قد قمت بكاتبة معلومات اكثر من اللازم لذا كان من الواجب عليك ان تغير بعض من هذه المدخلات اي انه اصبح من الواجب عليك ان تمسح الاشاياء الخطأ الموجودة عندك في صندوق العرض لذا فأن لغة البيرل قد وفرت لنا القابلية على المسح من داخل صندوق العرض وكما يلي في هذا الكود الذي يوضح كيف تتم عملية المسح

*CODE(61)

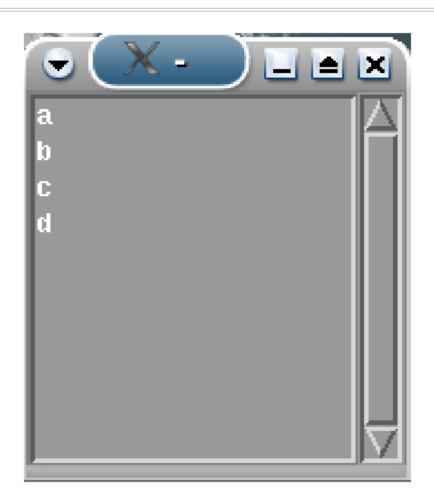
```
use Tk;
$top=MainWindow->new;
$a=$top->Scrolled("Listbox",-selectmode            =>'single',
                    -scrollbars      =>"e",
);
$a->insert('end',
"a",
"b",
"c",
"d",
"e",
"f",
"g",
"h",
"i",
);
$a->delete(
4,'end'
);
$a->pack;
MainLoop;
```

هذا البرنامج واضح ولايحتوي على اي شئ جديد او معلومة لم تمر علينا سوف الفقرة الاخيرة و هي الفقرة الاتية في الكود الاتي

*CODE(62)

```
$a->delete(
4,'end'
);
```

حيث ان الية المسح تتم كما يلي انه البيرل تعمل اولا على تحديد الرقم الذي قمت بأعطأه ثم تعمل على عد اربع اسطر ويقوم بعرضها ومة ثم يبدأ المسح من السطر رقم خمسة واليكم الصورة الخاصة بتنفيذ البرنامج



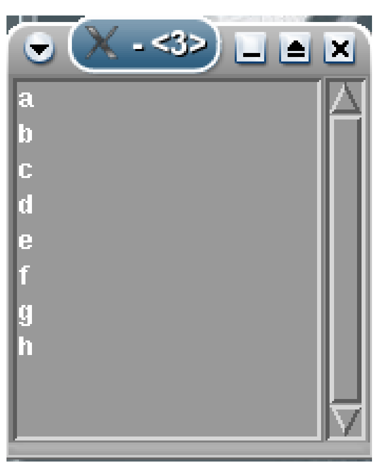
FIGURE(33)

هذا اسلوب من اساليب استعمال هذه الخاصية اما الاسلوب الاخر هو من الممكن ان تقوم بمسح العنصر الاخير فقط من البرنامج وتتم العملية كما يلي في هذا الكود

CODE(63)

```
use Tk;
$top=MainWindow->new;
$a=$top->Scrolled("Listbox",-selectmode          =>'single',
                 -scrollbars     =>"e",
);
$a->insert('end',
"a",
"b",
"c",
"d",
"e",
"f",
"g",
"h",
"i",
);
$a->delete(
4,'end'
);
$a->pack;
MainLoop;
```

كما نلاحظ انه الفرق الوحيد الذي حصل في البرنامجين هو انه تم الغاء الرقم 4 اي الرقم الذي حددناه و ان هذه الخاصية محددة بشكل افتراضي ان لم يتم تحديد رقم ان تقوم بمسح العنصر الاخير من صندوق العرض و الصورة الخاصة بتنفيذ البرنامج هي كألاتي



*FIGURE(34)

نلاحظ الان من الصورة الخاصة بالبرنامج انه تم مسح السطر الخاص بالعنصر الاخير

تحديد العناصر من خارج صندوق العرض

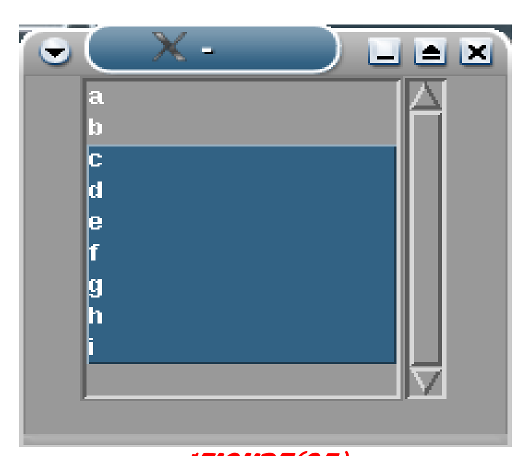
كما واضح انه من المعلوم لدينا انه من الممكن ان نختيار الملعومات التي موجودة لدينا في صندوق العرض من خلال النقر عليها وهذا يتم من خلال الامر(*selectmode*)

وهذا الامر وملحقاته قد شرحناها قبل قليل ولكن ما الذي يحصل اذا اردت ان تأشر على الاسطر الموجودة لديك في صندوق العرض من داخل البرنامج ومن دون استعمال الماوس وهذه العملية تتم كما يلي في هذا الكود

*CODE(64)

```
use Tk;
$top=MainWindow->new;
$a=$top->Scrolled("Listbox",-selectmode              =>'single',
                        -scrollbars      =>"e",
);
$a->insert('end',
"a",
"b",
"c",
"d",
"e",
"f",
"g",
"h",
"i",
);
$a->selectionSet(2,'end');
$a->pack;
MainLoop;
```

نلاحظ ان هذا البرنامج من ناحية العمل يشبه نوعا ما و الى حد ما برنامج الذي من خلاله بعملية المسح حيث يعمل على وفق نفس الاستراتيجية و اليكم الصورة الناتجة من تنفيذ البرنامج



*FIGURE(35)

حيث في هذا البرنامج ايضا يعمل على عد سطرين ويعمل على تظليل الباقي

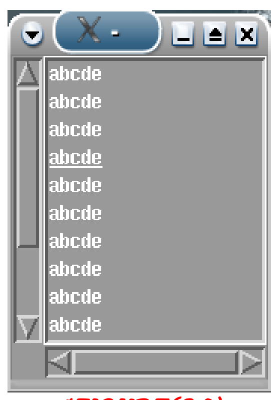
How to activate the elements on the list box

ان تقنية صندوق العرض في لغة البيرل تسمح لنا بان يتم تفعيل العنصر الذي تريد ان تقوم بتفعيله من خلال استعمال خاصية التفعيل وذلك كما يلي في الكود الاتي

*CODE(65)

```
use Tk;
$top=MainWindow->new;
$a=$top->Scrolled("Listbox",-selectmode   =>'e');
$a->insert('end',
"abcde",
"abcde",
"abcde",
"abcde",
"abcde",
"abcde",
"abcde",
"abcde",
"abcde",
);
$a->activate(3);
$a->focus;
$a->pack;
MainLoop;
```

ان هذا الكود قمنا من خلاله بتحديد ان العنصر الذي نريد ان نقوم بتفعليه هو العنصر الثالث فأن الناتج من هذا الكود هو انه يتم عد 3 عناصر وتفعيل السطر الرابع و الناتج من تنفيذ هذا البرنامج هو الصورة الاتية



*FIGURE(36)

Text Technic

الان من بعد ان انتهينا من برمجة كل ما يتعلق ببرمجة السكرول بار بصورة مفردة و صندوق العرض والسكرول بار بشكل مزدوج حان الوقت لكي نقوم بالانتقال الى تقنية جديدة هي هي تقنية النص
وان تقنية النص تعتبر من اهم التقنيات التي ومن اكثرها قوة لما تحتويه من قدرات وامكانيات كما سوف نلاحظ على طول الصفحات القادمة
سوف نأخذ الان كود مثالي يوضح كيف يتم تكوين منصة ال(*text*)
واليكم الكود

CODE(66)

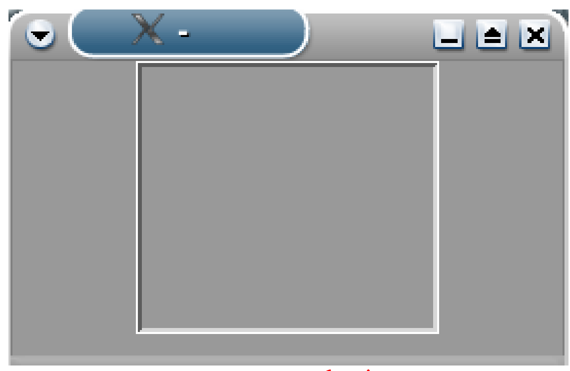
```
use Tk;
$top=MainWindow->new;
$top->Text(-width => 20, -height => 10)->pack;
MainLoop;
```

لاحظنا من خلال هذا البرنامج ان السطر المسئول عن تكوين منصة النص هو السطر التالي

CODE(67)

```
$top->Text(-width => 20, -height => 10)->pack;
```

هو هذا السطر المسئول عن تكوين منصة النص و الان لو نفذنا البرنامج لكان الناتج منه هو الصورة الاتية



FIGURE(37)

الان بعد ان اخذنا كود بسيط يوضح عمل منصة النص الان سوف ننتقل الى الخصائص والميزات المتقدمة الخاصة بتقنية النص

Advanced feature of text widget

من الخصائص المتقدمة لتقنية النص في لغة البيرل هي انه تحتوي على عدد كبير من الخصائص ومن هذه الخصائص هي

الخاصية الاولى
من الخصائص التي شرحناها ومن الممكن ان يتم تطبيقها في تقنية النص هي

1-relief
2-background
3-foreground
4-insertontime
5-insertofftime
6-state
7-font
8-high
9-width
10-selectbackground
11-selectforeground

جميع هذه التقنيات قد سبق وان قمنا بشرحها والان نعلم ماهي اساليب عملها وكيف من الممكن ان يتم ادخال المتغيرات و الخيارات التي تكون ملحقة بها ولذلك ايضا شوف لن اقوم بشرحها

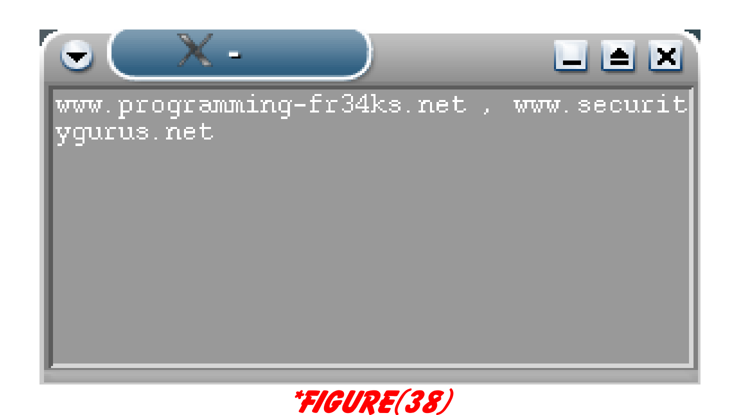
اما عن التقانات الاخرة فسوف نستعرض اغلبها من خلال الكوادت التالية اما الان فسوف نلتحق بتقنية جديدة تشبه هذه التقنية هي تقنية ال

الخاصية الثانية
الان من خلال هذه التقنية سوف نقوم باستخدام الطرق الخاصة بادخال النص الى صندقق التكست وان عملية ادخال النص تتم كانت تتم في تقنية صندوق العرض من خلال الامر(*insert*)
واليكم كيف تتم العملية كما في هذا الكود

CODE(68)

```
use Tk;
$top=MainWindow->new;
$a=$top->Text(-height          =>10,
              -width      =>40,
              );
$a->insert("end",
"www.programming-fr34ks.net , www.securitygurus.net",
);
$a->pack;
MainLoop;
```

وان ناتج تنفيذ هذه العملية يكون كما في الصورة الاتية



FIGURE(38)

العملية بسيطة وتتم كما كانت تتم من خلال صندوق العرض
الان انتهينا من بعد شرح تقنية النصوص الان سوف ننتقل الى تقنية اخرى مشابهة لهذه العملية

TagConfigure

هذه التقنية تعطيك اسلوب اخر من اساليب عنونة النصوص في منصات النصوص وهذه التقنية تقوم بمهمات تغير مظهر النص وتغير لون النص وتلوين حجم النص يعني يمكن القول انها تعمل على تغيير خصائص النص
الان سوف نأخذ بعض البرامج المختلفة لكي نرى امكانيات هذه التقنية

البرنامج الاول
اليكم البرنامج ومن ثم نناقش اسلوب عمله

*CODE(69)

```
use Tk;
$top=MainWindow->new;
$a=$top->Text();
$a->tagConfigure('bold',	-font	=>'arial 16 bold');
$a->insert("end","spawn is perl programmer ,\n",'bold');
$a->insert("end","spawn is perl programmer");
$a->pack;
MainLoop;
```

هذا البرنامج هو برنامج جميل وان كان يدل على شئ هو يدل على مدى مرونة القدرات الملحقة مع لغة البيرل
والان نأتي الى شرح البرنامج
الخطوة الاولى

*CODE(70)

```
$a=$top->Text();
```

هذه الخطوة هي عبارة عن الخطوة التي تكون مسئولة عن تكوين المنصة الخاصة بتقنية ال(text)
هذه هي عمل الخطوة الاولى
الخطوة الثانية

*CODE(71)

```
$a->tagConfigure('bold',	-font	=>'arial 16 bold');
```

هذه الخطوة عملها هي انها تعمل على تعريف ال(text tag)
اما عن المعلومات الموجودة من داخل البلوك البرمجي هي
اولا تحديد نمط الخط وقدد حددناه هنا من النمط العريض
ثانيا تحديد اسم الخط وهذا تم من خلال تحديد اسم الفونت الذي ترغب به
الخطوة الثالثة و الخطوة الرابعة كلاهما نفس الشئ وسبق ان شرحنا عمل اداة الاضافة(insert)
ولكن الذي احب اضيفه انه في الجملة الاولى تم اضافة كلمة(bold)
حتى مترجم البيرل يعرف اه هذه الجملة يجب ان تخرج بالنمط العريض عندما يتم تنفيذ البرنامج ولان
وبعد ان تم تنفيذ البرنامج اليكم الصورة الخاصة بتنفيذ البرنامج



*FIGURE(39)

ملاحظات على طرق الكاتبة
في هذه التقنية من اسلوب الكاتبة في لغة البيرل يوجد لدينا خاصيتان هما خاصيتان معروفتان ولكن لم يمرا علينا في الطرق السابقة ولذا حبيت ان اوضحهم
وهاتان الخاصيتان هما

1-underline
2-overstrike

واليكم الكود الخاص بهاتان الخاصياتان

*CODE(72)

```
use Tk;
$top=MainWindow->new;
$a=$top->Text();
$a->tagConfigure('overstrike',-font  =>'arial 16 overstrike');
$a->tagConfigure('underline',,-font  =>'arial 16 underline');
$a->insert('end',"spawn is perl programmer\n",'overstrike');
$a->insert('end',"spawn is perl programmer",'underline');
$a->pack;
MainLoop;
```

نأتي الى شرح البرنامج
الفقرة الاولى

*CODE(73)

```
$a->tagConfigure('overstrike',-font  =>'arial 16 overstrike');
```

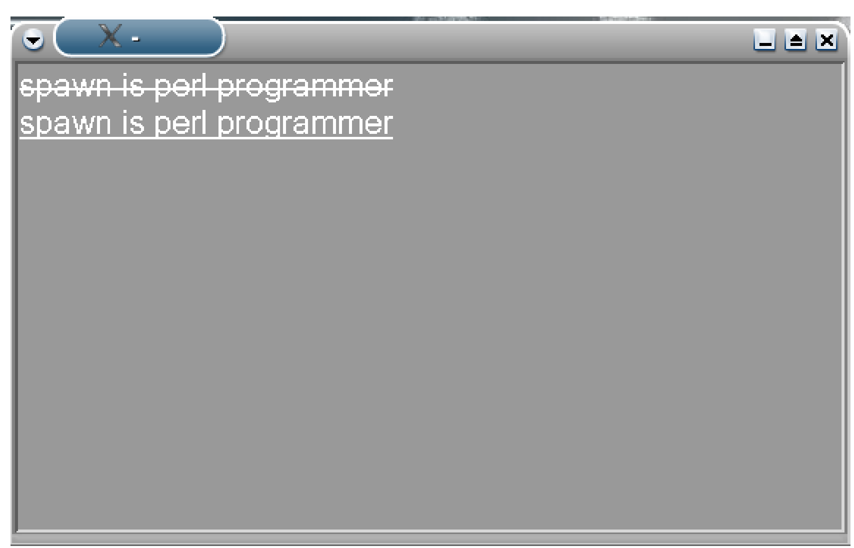
هذه الفقرة البرمجية تكون مسئولة عن وضع خط خلال النص الذي نقوم بكتابته
الفقرة الثانية

*CODE(74)

```
$a->tagConfigure('underline',,-font  =>'arial 16 underline');
```

الفقرة هذه مسئولة عن كتابة النص ويكون نوع النص من النوع الذي تحته خط
الفقرة الثالثة و الرابعة هما نفس الفقرة

حيث نقوم من خلال الامر المسئول عن ادخال النصوصو وهو الامر(insert)
حيث نقوم بادخال النص ومن ثم تذكر ثم تذكر انه يجب عليك ان تقوم بادخال المتغير الذي عملت عليه كما هو الحال مع نهاية هاتين الفقريتن
الان وبعد ان شرحنا هذا البرنامج وما يحتويه من فقرات حان الوقت لكي نقوم بتنفيذه و اليكم الصورة الخاصة به



*FIGURE(40)

الان بقيت في هذا الموضوع تقريبا فقرتين احب ان انوه اليها الفقرة الاولى وهبي متعلقة بالخيارات التي لها صلة بالالوان حيث ان الذي اريد من هذه الفقرة هو فقط جلب الانتباه الى ما يحصل في خيارات الالوان المتعلقة بهذه التقنية حيث انها لها اسلوب عمل مختلف
اليكم الكود الخاص بهذه الفقرة وهو الكود الاتي

*CODE(75)

```
use Tk;
$top=MainWindow->new;
$a=$top->Text();
$a->tagConfigure('background',-background       =>'blue');
$a->insert("end","It`s LinuX It`s The Open World It`s Freedom",'background');
$a->pack;
MainLoop;
```



*FIGURE(41)

الان على كل عندما نأتي الى تنفيذ البرنامج يكون ناتج تنفيذ البرنامج كما في الصورة التي في الاعلى استعمال الالوان في هذه الخاصية يعمل على تحديد النص المكتوب فقط باللون المحدد وليس كل المنصة لانه هذا الخيار هو خيار خاص بال(*tagConfigure*)
وليس خيار خاص بمنصة ال(*text*)

How to use ur own variable

من الممكن ان نقول ان هذا الموضوع هو من اهم المواضيع الموجودة في هذه التقنية بسبب انه من الممكن ان تقوم انت باستعمال متغيرات انت الذي تقوم بتحدديها وذلك لانه هذه الخاصية لها اسلوب خاص في العمل وهو اسلوب فريد بها لايتم استعماله في اساليب و تقنيات اخرى
الان سوف نأخذ كود من الكودات التي سبق وان قمنا بشرحها لكي نفهم مالذي تقوم به هذه الطريقة

*CODE(76)

```
use Tk;
$top=MainWindow->new;
$a=$top->Text();
$a->tagConfigure('background',-background        =>'blue');
$a->insert("end","It`s LinuX It`s The Open World It`s Freedom",'background');
$a->pack;
MainLoop;
```

هذا الكود هو نفس الكود الذي قمنا باستخدامه في الكود البرنامج السابق لغرض فهم عملية تغيير الخلفية الخاصة بالبرنامج اما الان الذي نريده من هذا البرنامج هو مختلف تماما
الان لنركز على هذه الخطوة الموجودة في هذا الكود

*CODE(77)

```
$a->tagConfigure('background',-background        =>'blue');
$a->insert("end","It`s LinuX It`s The Open World It`s Freedom",'background');
```

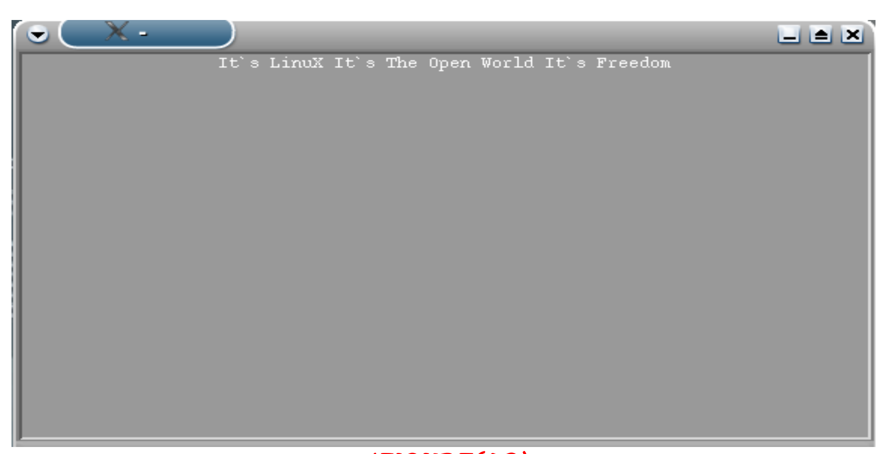
في الخطوة الاولى من البرنامج نلاحظ انه قد قمنا بتعريف ال(tagconfigure)
حيث حددنا في البداية كلمة ال(background)
ومن ثم استعملنا الامر(background)
ومن ثم قد قمنا باعطأه قيمة اللون الازرق
الى حد الان لم يطرأ شئ غريب
الخطوة الثانية هي
استعملنا امر الادخال الخاص بالنص من خلال خاصية الادخال(insert)
ومن ثم حددنا النص الذي نريد ان نقوم بادخاله ولكن بعد ان حددنا النص وضعنا فارزة ومن ثم ايضا وضعنا كلمة(background)
على كل اذا لاحظت انه كل من كلمتي ال(background)
في كلتا الحالتين كانتا هذان الكلمتان في وضع ال(string)
اي كانت موجودة بين علامات الاقتباس
ولك واحد يعرف انه في لغة البيرل كل كلمة مهما كانت عندما تكون موجودة بين علامات الاقتباس تكون سلسلة نصية
ما ارمي اليه من هذه النقطة هو انه ماذا يحدث لو غيرنا كلمة(background)
في كلتا الحالتين الى كلمة
linux
spawn
perl
او الى اي شئ اخر مالذي سوف يحدث ؟؟

بالتاكيد لاشئ لذا فأنه من الممكن لك ان تغير هاتين الكلمتين الى اي كلمة ترغبها كما في هذا الكود

*CODE(78)

```
use Tk;
$top=MainWindow->new;
$a=$top->Text();
$a->tagConfigure("spawn",-justify            =>center);
$a->insert("end",
"It`s LinuX It`s The Open World It`s Freedom","spawn");
$a->pack;
MainLoop;
```

قبل ان نتكلم عن هذا البرنامج سوف نقوم بتنفيذه ونرى ناتج تنفيذه و الان اليكم صورة البرنامج



*FIGURE(42)

حان الوقت لكي نتكلم عن البرنامج

*CODE(79)

```
$a->tagConfigure("spawn",-justify            =>center);
```

قمنا في هذه الخطوة بتعريف ال(tagconfigure)
اليكم ما هي اهمية كلمة(spawn)
في هذه الخطوة انا من ناحيتي اعتبرها بمثابة مخزن تطبيقي محصور بين علامات اقتباس حيث يتم فيه وضع الخيارات التي ترغب في تطبيقها على النص
ومن ثم من خلال امر الادخال(insert)
نقوم بادخال النص الذي نرغب فيه ومن ثم بعد ذلك نقوم بعملية استدعاء للمتغير الذي استعملناه لكي يتم تطبيق الاوامر او الخيارات التي نرغب في ان يتم تطبيقها على النص
ملاحظة هذا الكلام الذي ذكرته عن(spawn)
هو ليس كلام خاص بل كلام عن كل عن كل المتغيرات الموجودة في البرامج الموجودة في الاعلى
وملاحظة اخرى عن الخيار الذي يحمل الاسم(justify)
هذا الخيار هو خيار خاص بترتيب النص الذي تقوم بادخاله وهو خيار ب 3 احتمالات وهي كالتالي

1-right
2-center
3-left

الاول مسئول عن عرض النص من جهة اليمين
الثاني مسئول عن عرض النص في منطقة المنتصف
الثالث مسئول عن عرض النص من جهة اليسار

Scale

تقنية ال(scale)

او ما يعني في العربية مقايس التدرج او المقياس على كل هذه التقنية تشبه نوعا ما تقنية السكرول بار ولكن السكيل لاتعمل سكرول لشي اي انها لاتحرك النوافذ الى الاعلى او الى الاسفل مثل نوافذ السكرول بار ولكنم الذي تعمله هو ان تعمل سكرول لنفسها

How to create a scale

الان سوف نأخذ برنامج لعمل(scale)

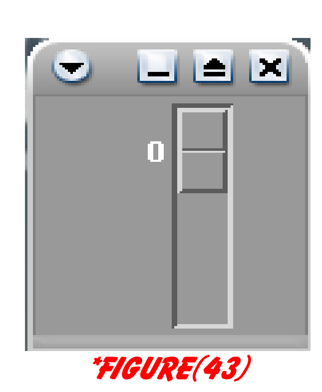
نموذجي اليكم الكود الخاص بهذه العملية

*CODE(80)

```
use Tk;
$top=MainWindow->new;
$a=$top->Scale();
$a->pack;
MainLoop;
```

كود بسيط لا يحتاج الى شرح عمله كل ما يقوم به هو تكوين(scale)

لاأكثر و لا اقل اليكم الصورة الخاصة بتنفيذ البرنامج



*FIGURE(43)

Scale options

ان تقنية ال(*scale*)
حالها كحال التقنيات الاخرى حيث انها لها العديد من الخصائص التي تجعلها لها القدرة على القايم باعمالها
بمرونة اكثر

الخاصية الاولى
هنا مثل كل مرة سوف نضع الخواص المشتركة مع الخواص الاخرى و التي سبق و ان قمنا بشرحها

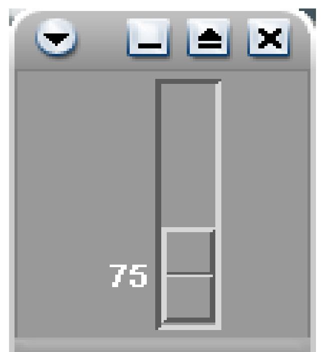
1-font
2-relief
3-background
4-foreground
5-label
6-orient
7-width
8-horizontal
9-state
10-highlightbackground
12-highlightthickness
13-highlightcolor

الخاصية الثانية
في هذه الخاصية سوف نعمل على تحديد المعيار الخاص للتقنية ال(*scale*)
والذي نريده منها ان تعمله يعني هي مبرمجة افتراضية ان تبدأ من الصفر الى حد المئة ولكن اذا اردنا ان نغير
هذه الخاصية يتم التغيير من خلال هاتين الخاصيتان كما في هذا الكود

CODE(81)

```
use Tk;
$top=MainWindow->new;
$a=$top->Scale(-from      =>0,-to    =>    75);
$a->pack;
MainLoop;
```

ومن خلال هذا الكود قمنا بتحديد نقطة البداية من خلال الخيار(*from*)
وحددنا النهاية من خلال الامر(*to*)



FIGURE(44)

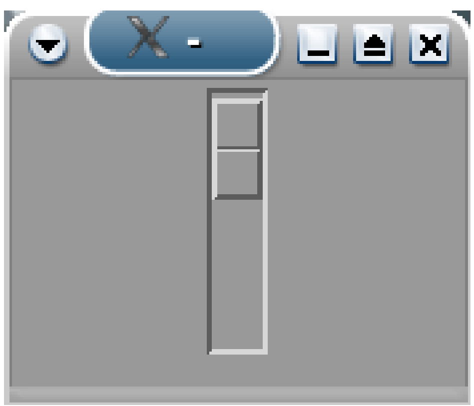
وكانت نقطة البداية من الصفر وكانت نقطة النهاية هي 75
واليكم الكود الخاص بهذه العملية

الخاصية الثالثة
طبعا هذه الخاصية من افتراضية مبرمجة اي انه من مجرد ان تنفذ الكود الذي يحتوي على تقنية ال (Scale)
فأن هذه القيمة تكون مفعلة ولكن هنا ألان لست بصدد شرح ماهو دورها اذا كانت مفعلة و لكن ماهو دورها اذا كانت معطلة وهذه الخاصية هي خاصية ال(showvalue)
واليكم الخاص بتعطيل هذه الخاصية

CODE(82)

```
use Tk;
$top=MainWindow->new;
$a=$top->Scale(-from        =>0,-to  =>100,-showvalue  =>0);
$a->pack;
MainLoop;
```

الان عندما نقوم بتنفيذ البرنامج نلاحظ الصورة الناتجة من تنفيذ البرنامج هي الاتي



FIGURE(45)

اذن نلاحظ من خلال صورة البرنامج انه القيمة التي كانت تظهر بجوار السكيل اختفت اي انه هذه الخاصية هي تكون مسئولة عن اظها ر هذه القيمة الي عندما تكون 1 تظهر القيمة الخاصة بال(scale)
واذا كانت القيمة صفر كما هو الكود السابق سوف تؤدي الى اختفاء القيمة كما لاحظنا

الخاصية الثالثة

هذه الخاصية تعمل على تقسيم ال(scale)

وفق تقاسيم رقمية انت تحددها وفق ما تريد لكي تعرف في اي موضع الفكرة لهذه الخاصية تصل افضل من خلال الكود الاتي

*CODE(83)

```
use Tk;
$top=MainWindow->new;
$a=$top->Scale(-from        =>0,-to  =>1000,-showvalue        =>1,-tickinterval=>100,-length
        =>1000,-orient        =>'horizontal');
$a->pack;
MainLoop;
```

هذا الكود كما هو واضخ يحتوي على عدد لابأس به من المعلومات داخل البلوك البرمجي الخاص به وشرح الخيارات هي

-from

تحديد نقطة البداية

-to

تحديد نقطة البداية

-showvalue

اظهار القيمة بجانب ال(scale)

-tickinterval

وهو الخيار الذي كنا نتحدث عليه وهو الخيار المحدد لهذه الخاصية حيث نلاحظ انه ال(scale)
ونلاحظ انه بجانب هذا الخيار يوجد
رقم ال 100
ونفس الوقت انه لدينا نقطة البداية من 0 الى 1000 ويوجد تحت ال(scale)

ارقام اخرى
تكرر كل زاد الرقم مقدار 100 اي انه هذا الخيار الخاص بتقسيم ال(Scale)
الى عشر مناطق كل وذلك لانه هذا الخيار يحمل الرقم مئة ولو كان يحمل رقم خمسين مثلا كان الرقم الذي يتكرر تحت ال(scale)
هو خمسين اذن وظفته هو تقسيم ال(scale)
الى مناطق يتم تحدديها من خلال هذا الخيار

-orient

موقع ال(scale)
افقى او عمودي

-length

الطول الخاص بال(scale)
واليكم الصورة الخاصة بهذا الخيار



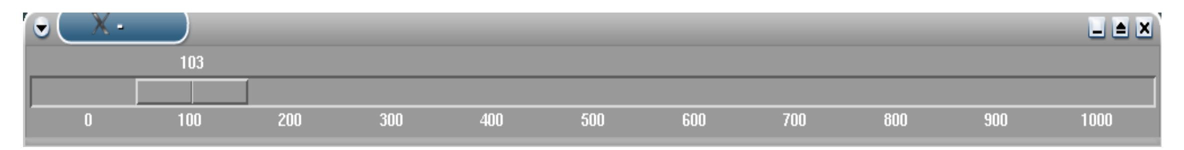
*FIGURE(46)

الخاصية الرابعة
هذه الخاصية خاصة بال(slider)
الخاص بال(scale)
وكيف يتم تغييرطوله كما تريد و اليكم الكود الخاص بهذه العملية

*CODE(84)

```
use Tk;
$top=MainWindow->new;
$a=$top->Scale(-from         =>0,-to  =>1000,-showvalue          =>1,-tickinterval=>100,-length
          =>1000,-orient         =>'horizontal',
-sliderlength=>100);
$a->pack;
MainLoop;
```

الصورة الخاصة بتنفيذ الكود هي



*FIGURE(47)

طبعا من النظر نلاحظ كم تغير طول ال(sliderbar)
بالمقارنة من البرنامج الماضي علما في الحالتين قمنا بتنفيذ نفس البرنامج

Frames MainWindows & Toplevels

في هذا الموضوع نقوم بمناقشة ما يتعلق بألاطارات و ال(*toplevels*)
طبعا الاطارات في لغة البيرل وضعها الافتراضي هو انه لايوجد تعبير بصري لها اذا تركت على وضعها الافتراضي لذا فيما لو اذا اردت ان تعمل اطأر ويكون له تأثير بصري فأنه لابد لك ان تعمل على تغيير الاعدادات الافتراضية الخاصة بألاطارات

والان اليكم البرنامج الخاص بهذه التقنية و هو كما يلي

***CODE(85)**

```
use Tk;
$top=MainWindow->new;
$a = $top->Frame(-borderwidth => 2, -relief => 'groove');
$a->Button(-text =>'hello world')->pack;
$a->Button(-text => 'hello world')->pack;
$a->pack;
MainLoop;
```

نلاحظ تقنية جعل الاطار مرئي هي خاصية ال(*releif*)
والصورة الناتجة من تنفيذ هذا الكود هي



***FIGURE(48)**

Toplevels

لكي تقوم بعملية استدعاء ال(toplevel)
فأن عملية الاستدعاء الخاصة بهذه التقنية لا تتم بصورة مباشرة بل تتم من خلال منصة يمكن القول
انها المنصة الاولى التي من خلالها يتم استدعاء ال(toplevel)
وعملية استدعاء ال(toplevel)
تتم كما يلي في هذا الكود

*CODE(86)

```
use Tk;
$top= MainWindow->new;
$top->title("MainWindow");
$top->Button(-text => "Toplevel", -command => \&do_Toplevel)->pack( );

MainLoop;
sub do_Toplevel {
  if (! Exists($tl)) {
    $tl = $top->Toplevel( );
    $tl->title("Toplevel");
    $tl->Button(-text => "Close",
            -command => sub { $tl->withdraw })->pack;
  }
}
```

والان نأتي الى شرح الفقرة الاولى من هذا البرنامج وهي كما يلي

*CODE(87)

```
$top->Button(-text => "Toplevel", -command => \&do_Toplevel)->pack( );
```

هذه الفقرة بسيطة من خلالها يتم تكوين زر عادي يكون النص الذي يحمله هو(toplevel)
ومن خلال استعمال الامر الخيار(command)
الذي نلاحظ انه يقوم باستدعاء روتين فرعي ومن ثم ينتهي عمل هذه الفقرة من البرنامج على كل يعني
هذه الفقرة من البرنامج فقرة بسيطة اما الان نأتي الى فقرة الثانية وهي

*CODE(88)

```
sub do_Toplevel {
  if (! Exists($tl)) {
    $tl = $top->Toplevel( );
    $tl->title("Toplevel");

  }
}
```

شرح هذا الرويتن الفرعي هو كما يلي
اذا كان المتغير الذي نعمل عليه موجود فأنه سوف يتكون لدينا ما يلي

CODE(89)

```
$tl = $top->Toplevel( );
```

ستتكون نافذة جديدة من نوع ال(toplevel)
ثم بعد ذلك يكون

CODE(90)

```
$tl->title("Toplevel");
```

ثم ان هذه النافذة التي تكونت سوف تحمل الاسم(toplevel)
الخطوة الاخرى

CODE(91)

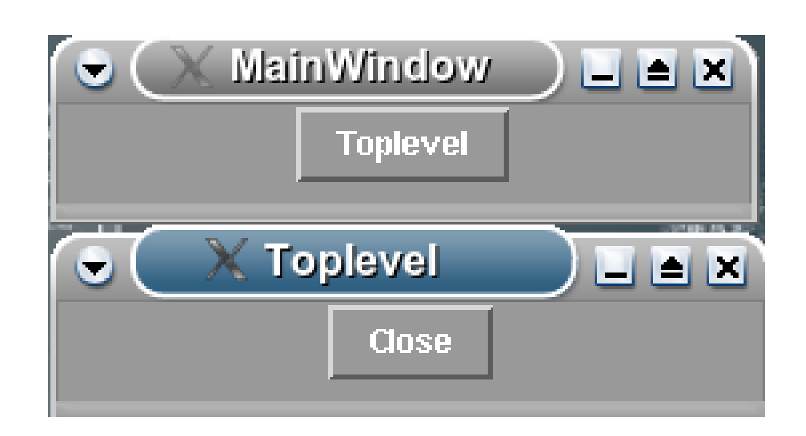
```
$tl->Button(-text => "Close",
        -command => sub { $tl->withdraw })->pack;
```

الان سوف يتكون لدينا زر عادي اخر يحمل كلمة اغلاق على سطحه ومن ثم استعمال الامر او الخيار كوماند حيث نلاحظ انه يحمل روتين فرعي اهمية هذا الرويتن الفرعي سنشرحها بعد قليل
الان سوف نرى ما هي الصورة الناتجة من تنفيذ البرنامج



*FIGURE(49)

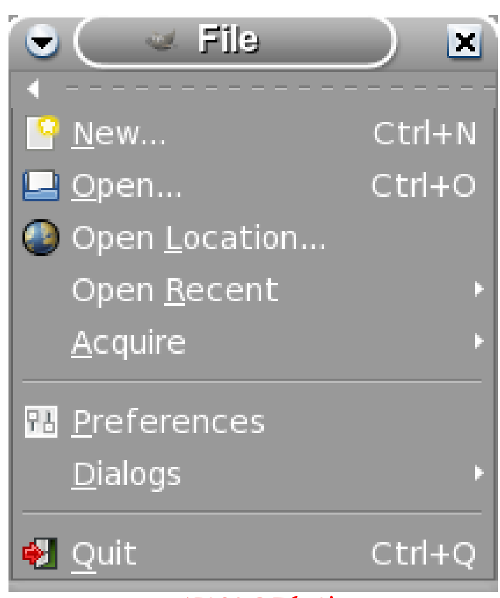
الان هذه الصورة الاولى من البرنامج اي انه هذه الصورة الخاصة بالبرنامج قبل ان يتم استدعاء ال(toplevel)
الان سنرى صورة البرنامج من بعد ان يتم استدعاء ال(toplevel)



*FIGURE(50)

هذه هي صورة البرنامج بعد ان تم استدعاء ال(toplevel)
اي ان القسم الثاني من الصورة هو ال(toplevel)
وهذا ما نلاحظه من خلال اسم البرنامج
على كل الان سنرجع قليلا الى الوراء وهي ان خيار الكوماند كان مسند اليه قيمة روتين الفرعي
نلاحظ في نافذة ال(toplevel)
يوجد زر الاغلاق واذا قمنا بالضغط على زر الاغلاق ستم اغلاق نافذة ال(toplevel)
لوحدها اذن هذه وظيفة الرويتن الفرعي الموجود في ال(toplevel)
وهنالك من يقول لماذا لانقوم باسناد الروتين الفرعي(exit)
الى خيار الكوماند بدلا من الخيار المسند اليه وهو ال(with_draw)
ولكن الفرق بين هذين الروتينين الفرعيين هو انه الاول اي ال(exit)
يعمل على اخراجك من البرنامج كله اي يتم اغلاق البرنامج
اما الرويتن الثاني فأنه يعمل على اغلاق ال(toplevel)
لوحدها والنافذة التي تم استدعاء ال(toplevel)
منها سوف تبقى موجودة ولايتم اغلاقها

هذا الجابتر من الكتاب يشرح ماهي ال(*menu item*)
وكيف تتصرف وكيف تكون وماهي محتوياته اضافة الى بعض الخصائص المتعلقة بهذه التقنية
ولمن لايعرف ماهي ال(*menu item*)
هاهي في الصورة الاتية و لو انا متأكد انه الجميع يعرفها



*FIGURE(51)

هذه هي ال(*menu*)
التي اتكلم عنها وعلى كل هذه القائمة الخاصة ببرنامج الجمب الذي عندي
اضافة الى هذه القائمة سوف نناقش انواع اخرى من القوائم اضافة ال القوائم الكلاسيكية المعروفة على كل
سنتعرف اكثر عن القوائم مع الكودات الاتية في الصفحات التالية

How to create menu

في هذا البرنامج سوف نقوم بكتابة برنامج يقوم هذا البرنامج بتكوين قائمة بسيطة جدا لكي نعرف كيف تكون القائمة في البيرل على كل الكود الخاص بهذه العملية هو الكود الاتي

CODE(92)

```
use Tk;
$top=MainWindow->new;
$top->configure(-menu => my $menubar = $top->Menu);
my $file = $menubar->cascade(-label => '~File');
MainLoop;
```

شرح هذا الكود هو كالتالي الفقرة الاولى

CODE(93)

```
$top->configure(-menu => my $menubar = $top->Menu);
```

هذه الفقرة مكونة من الخيار

-menu

هذا الخيار يكون مسند الى متغيروهذا المتغير الذي تمت اليه عملية الاسناد له عمل مهم في الفقرة الثانية من البرنامج اما الجزء الثاني من هذه الخطوة فسوف اتركه لكي أشرحه بعد قليل
الفقرة الثانية من البرنامج هي

CODE(94)

```
my $file = $menubar->cascade(-label => '~File');
```

اما عن هذه الخطوة فأنها تكون مسئولة عن تكوين القائمة من خلال المتغير الذي اعتمدناه في الخطوة السابقة حيث من خلال المعرف(*cascade*)
نعرف ال(*label*)
الخاص بهذه القائمة الان ننفذ البرنامج ومن التنفيذ نحصل على الصورة الاتية



*FIGURE(52)

هذا هو الناتج من تنفيذ البرنامج السابق
ولكن نلاحظ ان هذا البرنامج نوعا ما تركيبته غير مفهومة وتحتوي على بعض التراكيب الجديدة لذا من الممكن ان نقوم باعادة صياغة البرنامج اعلاه بالشكل التالي ليكون كما يلي

*CODE(95)

```
use Tk;
$top=MainWindow->new;
$menubar=$top->Menu;
$top->configure(-menu        =>$menubar);
$file=$menubar->cascade(-label=>"~file");
MainLoop;
```

الان هذا البرنامج من الممكن ان نفهم ماهي محتوياته وايضا ممكن ان نعرف ممن ماذا يتكون ومن دون اي شرح مسبق لانه برنامج اسلوب كتابته هي نفس الاسلوب الذي تعملنا عليه
ومن المفيد في هذه الصيغة انه ناتج تنفيذها هو نفس ناتج تنفيذ البرنامج السابق ولك ان تتأكد من ذلك بنفسك

الان سوف نأخذ برنامج اخر ولكن متقدم قليلا حيث انه يحتوي على بعض الخصائص الاخرى
الخاصية الاولى
هذه الخاصية هي خاصية اضافة عنصر الى القائمة التي برمجناها و العملية تتم كما يلي اليكم الكود الخاص بها

***CODE(96)**

```
use Tk;
$top=MainWindow->new;
$menubar=$top->Menu();
$top->configure(-menu        =>$menubar);
$file=$menubar->cascade(-label              =>"~file");
$new=$file->cascade(-label =>"~New");
MainLoop;
```

الان لو قمنا في تنفيذ البرنامج سوف نلاحظ انه من خلال القائمة التي تحمل الاسم فايل سوف يتم
انسدال خيار اخر هو خيار ال(new)
و هذه الخيار الذي كان مسئول عن تكوينه

***CODE(97)**

```
$new=$file->cascade(-label =>"~New");
```

هنالك فقرة واحدة احب ان انبه اليها بشده و هي انه مهما كانت الخيارات التي او الاوامر التي ترغب في ان تقوم
باضافتها الى القائمة الواحدة يجب ان يكون التعريف الخاص بالمتغيرات يكون من خلال المتغير الذي تم اعتماده
في البداية لكي هو مسئول عن تكوين القائمة الاساسية يعني سوف اوضح الكلام على هذا الكود
يعني في برنامجنا السابق كانت القائمة الاساسية في البرنامج هي القائمة التي كانت تحمل الاسم(File)
واذا قمت بالكبس عليها فأن القائمة المنسدلة منها تحمل الاسم نيو اليس كذلك؟؟
اذن القائمة المنسدلة نيو هي قائمة معرفة داخل القائمة ملف لذا يمكن القول انه القائمة نيو هي قائمة
معرفة
داخل القائمة ملف ولهذا تم تعريف القائمة نيو من خلال استعمال المتغير
الذي كان قد تم استعماله من قبل لكي يتم تكوين القائمة ملف

***CODE(98)**

```
$new=$file->cascade(-label =>"~New");
```

ولذا انظر انه من خلال المتغير($new)
استعملنا المتغير($file)
لكي يكون مساهم في تكوين القائمة الجديدة نيو ولا يجوز ان يتم استعمال متغير اخر

الخاصية الثانية
و هذه الخاصية هي نوع من الخاصيات التي تعني بأمور الفصل بين ال(items)
الموجودة في القائمة التي برمجتها و هي تكون كالتالي

***CODE(99)**

```
use Tk;
$top=MainWindow->new;
$menubar=$top->Menu();
$top->configure(-menu       =>$menubar);
$file=$menubar->cascade(-label              =>"~file");
my $new = $file->cascade(
        -label      => 'New',
);
$file->separator;
$open=$file->cascade(
        -label          =>"Open",
                );
MainLoop;
```

من النظرة الاولى من البرنامج نلاحظ انه تقريبا لايحتوي على شي يختلف نوعا ما عن البرنامج السابق ولكن سوف نعمل(quick overview)

***CODE(100)**

```
$file=$menubar->cascade(-label              =>"~file");
```

تكوين القائمة الاولى او القائمة الاساسية و التي تحمل اسم(file)

الفقرة الثانية

***CODE(101)**

```
my $new = $file->cascade(
        -label      => 'New',
);
```

اما عن هذه الفقرة فهي تكوين القائمة الثانية التي تكون منسدلة من القائمة الاولى

الفقرة الثالثة

***CODE(102)**

```
$file->separator;
```

نرجع لهذه الفقرة بعد قليل

الفقرة الرابعة

***CODE(103)**

```
$open=$file->cascade(
        -label          =>"Open",
                );
```

هذه الفقرة هي تكون مسئولة عن تكوين القائمة المنسدلة الثانية من القائمة الاساسية الاولى

اما الان سنرى ما هو ناتج تنفيذ البرنامج لنرى مالذي سوف يحدث



*FIGURE(53)

هذا هو ناتج تنفيذ البرنامج ونلاحظ الان في الصورة يوجد خط فاصل بين القائمة الاولى ال(new)
والقائمة الثانية(file)
و هذا الخط الذي نتكلم عنه هو الخط الناتج من جراء استعمال هذا الكود

*CODE(104)

```
$file->separator;
```

هو الذي أدى ظهور هذا الخط

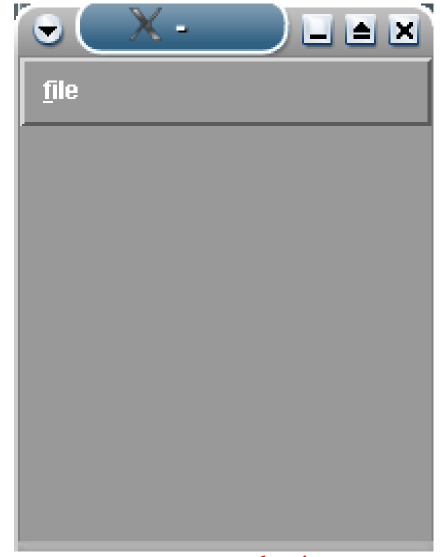
*HINT

عندي تلميح احب ان اشير اليه و هذا التلميح هو انه لا يظن البعض ان الناتج من تنفيذ الكودات الموجودة في الاعلى هو الاتي



*FIGURE(54)

ولكن الناتج من تنفيذ البرنامج هو يكون كما الاتي في هذه الصورة



*FIGURE(55)

وانا قمت بالفصل بينهم اي لا يظن البعض انه لديهم خلل برمجي

الخاصية الثالثة
هنا الان سوف نقوم بعمل توسع قليلا في القوائم ونأخذ اكثر من خاصية على كل اليكم هذا الكود

*CODE(105)

```
use Tk;
$top=MainWindow->new;
$menubar=$top->Menu;
$top->configure(-menu        =>$menubar);
$file=$menubar->cascade(
                -label          =>"File",
                        );
$new=$file->command(
                        -label          =>'New'
                );
$file->separator;
$open=$file->command(
                        -label          =>"Open",
                        -accelerator    =>"Ctrl-o",
                        -underline      =>0,
                        );
MainLoop;
```

الان سوف نعمل على مناقشة الامور الجديدة التي يحتويها البرنامج الجديد
الفقرة الاولى

*CODE(106)

```
$open=$file->command(
                    -label         =>"Open",
                    -accelerator   =>"Ctrl-o",
                    -underline     =>0,
                    );
```

هذه هي الفقرة الوحيدة الجديدة في البرنامج و التي لم نتطرق اليها من قبل
الجزء الاول من الفقرة الاولى

*CODE(107)

```
$open=$file->command
```

ان سبب استعمال هذه الامر هو لغاية القدرة على تطبيق بعض الامرو و الخصائص البرمجية التي لا نكون قادرين على ان نطبقها من خلال الامر

*CODE(108)

```
cascade
```

الجزء الثاني من الفقرة الاولى

*CODE(109)

```
-label          =>"Open",
```

هي الفقرة التي تكون مسئولة عن اعاطء الاسم للقائمة التي نعمل عليها
الجزء الثالث من الفقرة الاولى

*CODE(110)

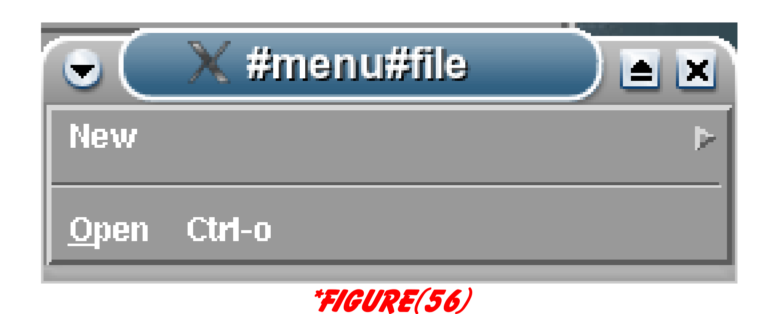
```
-accelerator    =>"Ctrl-o",
```

كما هو واضح انه هذه الفقرة تعمل على اعطاء(shortcut)
للقائمة التي برمجناها
الجزء الرابع من الفقرة الاولى

*CODE(111)

```
-underline      =>0,
```

هذه الفقرة تكون مسئولة عن وضع خط تحت الحرف الاول من القائمة
واليكم الصورة الناتجة من تنفيذ البرنامج و هي كما الاتي



*FIGURE(56)

هذه هي الصورة الخاصة الناتجة من تنفيذ البرنامج وايضا هنالك فقرة اخرى هي انه نحن لم نقوم ابدا باستعمال الامر

CODE(112)

```
title
```

اي ما أعنيه انه لم نستخدم الامر الخاص بوضع عنوان للبرنامج اذن عليك ان تعلم ان اسم هذا البرنامج هو تم وضعه(*by default*)

الخاصية الرابعة
في هذه القائمة سوف نقوم بعمل اضافة قائمة جديدة الى القائمة القديمة ومن خلال هذه العملية سوف تتضح لنا بعض الفقرات التي لم تتضح لنا من خلال البرنامج او البرامج الماضية
واليكم الكود الخاص بهذه العملية وهو كما الاتي

CODE(113)

```
use Tk;
$top=MainWindow->new;
$menubar=$top->Menu();
$top->configure(-menu       =>$menubar);
$file=$menubar->cascade(-label            =>"File");
$open=$file->command(
                -label              =>"New",
                -underline          =>0,
                -accelerator        =>'Ctrl-o'
);
$file->separator;
$new=$file->command(
                -label              =>"Open",
                -underline          =>0,
                -accelerator        =>'Ctrl-o'
);
$file->separator;
$exit=$file->command(
                -label       =>"Exit",
                -underline   =>0,
                -command     =>\&exit
                );
$edit=$menubar->cascade(
                -label       =>"Edit"
                );
$undo=$edit->command(
                -label       =>"Undo",
                -underline          =>0,
                -accelerator        =>'Ctrl-z'
                );
$edit->separator;
$redo=$edit->command(
                -label              =>"Redo",
                -underline          =>0,
                -accelerator        =>'Ctrl-y'
                );
MainLoop;
```

الان بعد ان قمنا بكتابة هذا الكود سوف نقوم بعمل اشرح الخاص لكي نعرف مالذي يجري داخل الكود

الفقرة الاولى
ماهو الفرق بين هذين المقطعين؟؟
هذا المقطع الاول

CODE(114)

```
$edit=$menubar->cascade(
                -label          =>"Edit"
                );
```

وبين هذا المقطع

CODE(115)

```
$redo=$edit->command(
                -label                =>"Redo",
                -underline            =>0,
                -accelerator          =>'Ctrl-y'
                );
```

وقبل ان تنظر الى هذين المقطيعن البرمجيين عليك ان تعرف انه كل من خيارات ال

1-label
2-underline
3-accelerator

جميع هذه الخيارات هي فعلا خيارات ولكنها ليست الخيارات التي نبحث عنها اذن في هذه الحالة لم يتبقى الا فرق واحد فقط وهو

CODE(116)

```
$edit=$menubar->cascade
```

CODE(117)

```
$redo=$edit->command
```

المقطع الاول يحتوي على كلمة(cascade)
والمقطع الثاني لا يحتوي على كلمة(cascade)
ولكن يحتوي على كلمة(command)
ماهو الفرق بين هاتين الكلمتين؟؟
نلاحظ في البرنامج السابق انه تم استعمال الامر

CODE(118)

```
cascade
```

مرتين فقط وفي المرتين التي تم استعماله فيها كان الذي يتم فيها استعماله يتم تكوين نافذة اساسية مثل نافذة الملف(file)
وبينما الامر

CODE(119)

```
command
```

هذا الخيار تم استعماله اكثر من مرة ولكن ايضا كل مرة تم استعماله فيها يتم تكوين قائمة جانبية يعني مثل قائمة ال(open,undo)
اذن من هذا الكلام نطلع بخلاصة هي انه كلما اردت ان تقوم بتكوين نافذة اساسية يتم استعمال الامر(cascade)

وكلما اردت ان تقون بتكوين قائمة منسدلة تقوم باستعمال الامر(command)
هذه هي اهم الفقرات التي كنت اريد ان اشرحها في البرنامج واليكم الان صورة تنفيذ البرنامج وهي كما الاتي



*FIGURE(57)

الخاصية الخامسة
الان في هذه الخاصية اضافة الى ما تم ذكره من الخصائص البرمجية في البرنامج السابق سوف نقوم بذكر كيفية اضافة جملة طباعة الى البرنامج لكي تقوم بطباعة المعلومات التي ترغب بها
ونلاحظ هنا انه قد ذكرنا انه توجد جملة طباعة وبما انه قد ذكرنا انه يوجد جملة الطباعة فأن الناتج من جملة الطباعة هو لن يظهر في البرنامج اي لن يكون الناتج من عملية الطباعة بصريا بل سوف يكون في الشيل واليكم الكود الخاص بهذه العملية وهو كألتالي

*CODE(120)

```
use Tk;
my $top = MainWindow->new;
$top->configure(-menu => my $menubar = $top->Menu);
my $file = $menubar->cascade(-label => '~File');
my $edit = $menubar->cascade(-label => '~Edit');
my $help = $menubar->cascade(-label => '~Help');
my $new = $file->cascade(
    -label       => 'New',
    -accelerator => 'Ctrl-n',
    -underline   => 0,
);
$file->separator;
$file->command(
    -label       => 'Open',
    -accelerator => 'Ctrl-o',
    -underline   => 0,
);
$file->separator;
$file->command(
    -label       => 'Save',
    -accelerator => 'Ctrl-s',
    -underline   => 0,
);
$file->command(
    -label       => 'Save As ...',
    -accelerator => 'Ctrl-a',
    -underline   => 1,
);
$file->separator;
$file->command(
    -label       => "Close",
    -accelerator => 'Ctrl-w',
    -underline   => 0,
    -command     => \&exit,
);
$file->separator;
$file->command(
    -label       => "Quit",
    -accelerator => 'Ctrl-q',
    -underline   => 0,
    -command     => \&exit,
);
$edit->command(-label => 'Preferences ...');
$help->command(-label => 'Version', -command => sub {print "This is the first edition of this
programm\n"});
```

```
$help->separator;
$help->command(-label => 'About',   -command => sub {print "this is a simple perl menu programm\n"});
MainLoop;
```

الان نلاحظ انه لو قمنا بتنفيذ البرنامج سوف نحصل على الصورة هذه



*FIGURE(58)

الان لو قمنا بفتح القائمة(*help*)
سوف نلاحظ انها تحتوي على قائمتان ثانويتنان هما(*About,version*)
واذا قمت باختيار كل القائمة(*about*)
سوف يتم طباعة هذه الجملة في الشيل

this is a simple perl menu programm

*FIGURE(59)

هذا يكون بالنسبة للقائمة الثانوية(*about*)
اما بالنسبة للقائمة الثانية(*version*)
عندما تقوم باختيارها سوف يتم طباعة الجملة هذه في الشيل

This is the first edition of this programm

*FIGURE(60)

وهذا الي يتم طباعته عندما يتم اختيار القائمة(*Version*)

الخاصية السادسة

في الخصائص السابقة قمنا بأضافة بعض المقاطع البرمجية التي كان من شأنها ان تعمل على ان تزيد من تطور القائمة التي نبرمج عليها واما الان سوف نضيف خيار اخر الى هذه القائمة التي نعمل عليها و الان في هذه

الخاصية سوف نعمل على تدعيم الخيارات في القائمة لكي تكون القائمة نموذجا حقيقيا من قوائم البرامج التي نعمل عليها على كل خاصية الخيارات التي تكلمنا عنها الان هي اضافة ال(*radio buttons*)
الى القائمة و اليكم الكود الذي يكون مسئول عن هذه العملية

*CODE(121)

```
use Tk ;
$top = MainWindow->new;
$top->configure(-menu => $menubar = $top->Menu);
$file = $menubar->cascade(-label => '~File');
$edit = $menubar->cascade(-label => '~Edit');
$help = $menubar->cascade(-label => '~Help');
$new = $file->cascade(
        -label       => 'New',
        -accelerator => 'Ctrl-n',
        -underline   => 0,
);
$file->separator;
$file->command(

        -label          => 'Open',
        -accelerator    => 'Ctrl-o',
        -underline      => 0,
        -command        =>\&favourite,
);
$file->separator;
$file->command(
        -label       => 'Save',
        -accelerator => 'Ctrl-s',
        -underline   => 0,
);
$file->command(
        -label       => 'Save As',
        -accelerator => 'Ctrl-a',
        -underline   => 1,
);
$file->separator;
$file->command(
        -label       => "Close",
        -accelerator => 'Ctrl-w',
        -underline   => 0,
        -command     => \&exit,
);
$file->separator;
$file->command(
        -label       => "Quit",
        -accelerator => 'Ctrl-q',
        -underline   => 0,
        -command     => \&exit,
);
```

```
$edit->command(-label => 'Preferences ...');

$help->command(-label => 'Version', -command => sub {print "This is the first edition of this programm\n"});
$help->separator;
$help->command(-label => 'About',   -command => sub {print "this is a simple perl menu programm\n"});
sub favourite {
$lang="perl";
foreach (qw/ perl c python ruby php/){
$top->Radiobutton(
        -text           =>"Choose ur favoutite language",
        -text           =>$_,
        -variable       =>\$lang,
        -value          =>$_,
        )->pack(-side  =>'left');
}
}
MainLoop;
```

هذا هو الكود الذي يكون مسئول عن هذه العملية الان نأتي الى شرح الفقرات الجديدة من البرنامج لكي نفهم مالذي يجري داخل البرنامج

الفقرة الاولى

*CODE(122)

```
sub favourite {
$lang="perl";
foreach (qw/ perl c python ruby php/){
$mw->Radiobutton(
        -text           =>$_,
        -variable       =>\$lang,
        -value          =>$_,
        )->pack(-side  =>'left');
}
```

هذه الفقرة هي روتين فرعي لانه
اولا قد تم اسنادها الى الخيار كوماند
ثانيا نلاحظ في بدايتها انها تحتوي على الكلمة الخاصة بتعريف الرويتن الفرعي وهي كملة(sub)
اما بعد ذلك تقريبا المحتوى البرمجي لهذا الروتين الفرعي يشبه المحتوى البرمجي للبرنامج الخاص بتلوين الراديو بوتن
على سوف نقوم بعمل مرور سريع لهذه الخيارات

1-

*CODE(123)

```
        -text           =>$_,
```

هذا الخيار عمله هو ان يقوم بترتيب الخيارات الموجودة ضمن البلوك الخاص بجملة الفور

2-

*CODE(124)

```
-variable       =>\$lang,
```

هذا الخيار هو الخيار الخاص بأن يتم اسناد قيم متغير اليه فيما لو اذا كان لدينا متغير في البرنامج ولا نريد ان يبقى على وضعه ان لا نرغب في أن يبقى متغير فكل ما نعمله ان نقوم بأسناده الى هذا الخيار

3-

*CODE(125)

```
-value          =>$_,
```

هذا الخيار هو الخيار الخاص بألتأشير على قيمة الزر حيث لو انه قد تم الغاءه فأن الذي يحصل عن ذلك هو انه سوف يحصل خلل في عمليات التأشير ويتم تأشير جميع القيم مرة واحدة اي تصبح جميع القيم مؤشرة او بمعني اخر جميع القيم قد تم اختيارها
اما الخيار الذي تم اسناده مع الامر(*pack*)
وهو الخيار(*left*)
هو مجرد خيار لكي يتم ترتيب اسماء لغات البرمجة بشكل جميل فقط اي ليس له وظيفة اخرى وانه من الممكن ان يستمر البرنامج من دونه ولكن وجودة يعطي جمالية للبرنامج فقط

اما الان وبعد هذا الشرح سوف نرى صورة البرنامج من بعد التنفيذ كيف تكون



*FIGURE(61)

الان هذه هي صورة البرنامج بعد ان قمنا باختيار القائمة الرئيسية ملف ومنها اخترنا القائمة الثانوية (open) ولكن على الرغم من الجهد الذي بذلناه يبدو البرنامج غير مقنع
لانه اي برنامج كان يحتوي على قائمة فأن المتفرعة من القائمة الثانوية تبدر ايضا على شكل قائمة ولا تبدو على شكل سطر كما هو الحال عنا الان على الرغم من ان البرنامج الذي كتبناه في الاعلى هو برنامج من البرمجية و من الناحية الهيكلية هو برنامج صحيح 100 في المئة ولكن شكله غير مقنع لانه قد تعودنا على انه القائمة المنبثقة من الفرعية تكون ايضا على شكل قائمة ولذا على هذا المعتقد سوف نعمل على ان نغير اوراقنا لكي نقوم بكتابة برنامج له قوائم مثل القوائم الخاصة بالبرامج التي نعرفها

الخاصية السابعة
في هذه الفقرة الان سوف نتكلم عن برمجة القائمة لكي تكون بشكل افضل ولكي تكون لدينا قائمة برنامج تكون اقرب الى القوائم الحقيقية التي نراها في البرامج التي نتعامل معها يوميا على كل اليكم الخاص بهذه القائمة ويكون كما يلي

*CODE(126)

```
use Tk;
$top=MainWindow->new;
$menubar=$top->Menu();
$top->configure(-menu        => $menubar);
$file=$menubar->cascade(-label          =>"File");
$edit=$menubar->cascade(-label          =>"Edit");
$help=$menubar->cascade(-label          =>"Help");
$open=$file->command(
                        -label          =>"Open",
                        -accelerator    =>"Ctrl-o",
                        -underline      =>0,
                        );
$new=$file->cascade(
                        -label          =>"New",
                        -menuitems      => [
['checkbutton',"spawn"],
['radiobutton',"perl"],
['radiobutton',"c"],
['radiobutton',"python"],
['radiobutton',"ruby"],
['radiobutton',"php"],
],
);
$close=$file->command(
                        -label          =>"Exit",
                        -accelerator    =>"Ctrl-e",
                        -underline      =>0,
                        -command        =>sub {exit},
                        );
$undo=$edit->command(
                        -label          =>"Undo",
                        -accelerator    =>"Ctrl-z",
                        -underline      =>0,
                        );
$redo=$edit->command(
                        -label          =>"Redo",
                        -accelerator    =>"Ctrl-y",
                        -underline      =>0,
                        );
$help->command(-label => 'Version', -command => sub {print "This is the first edition of this 
programm\n"});
$help->separator;
$help->command(-label => 'About',  -command => sub {print "this is a simple perl menu 
programm\n"});
MainLoop;
```

الان هذا هو الكود عن تكوين القائمة التي في تكوينها
ونأتي الى مناقشة الفقرات المكونة لها
الفقرة الاولى

*CODE(127)

```
$new=$file->cascade(
                        -label          =>"New",
                        -menuitems      => [
['checkbutton',"spawn"],
['radiobutton',"perl"],
['radiobutton',"c"],
['radiobutton',"python"],
['radiobutton',"ruby"],
['radiobutton',"php"],
],
);
```

الان لو نلاحظ البرنامج ككل نلاحظ انه هذه هي الفقرة الوحيدة التي لم تمر علينا في خلال مشوارنا مع برمجة القوائم على كل مكونات هذا المقطع البرمجي هي كألاتي
الجزء الاول

*CODE(128)

```
$new=$file->cascade(
```

هذا الجزء من البرنامج نلاحظ فيه على غير العادة انه يحتوي على الامر(cascade)
ولا يحتوي على الامر(command)
الذي تعودنا عليه في مثل هذه الاماكن من البرنامج والسبب في هذا انه هذه القائمة سوف تعتبر قائمة اساسية لما سوف يندرج عليها ولكن تبقى هذه القائمة قائمة فرعية للقائمة التي تحمل الاسم(file)
يعني سوف نوضح الفكرة كما يلي
القائمة الاساسية الرئيسية هي القائمة(file)
القائمة(new)
تعتبر قائمة فرعية من القائمة الاولى(file)
والقائمة الفرعية التي تنسدل من القائمة(new)
تعتبر قائمة فرعية من القائمة(new)
والقائمة(new)
هي القائمة الاساسية لها
هكذا يكون ترتيب البرمجي للقوائم في مثل هذه الحالات

الجزء الثاني

*CODE(129)

```
-label          =>"New",
```

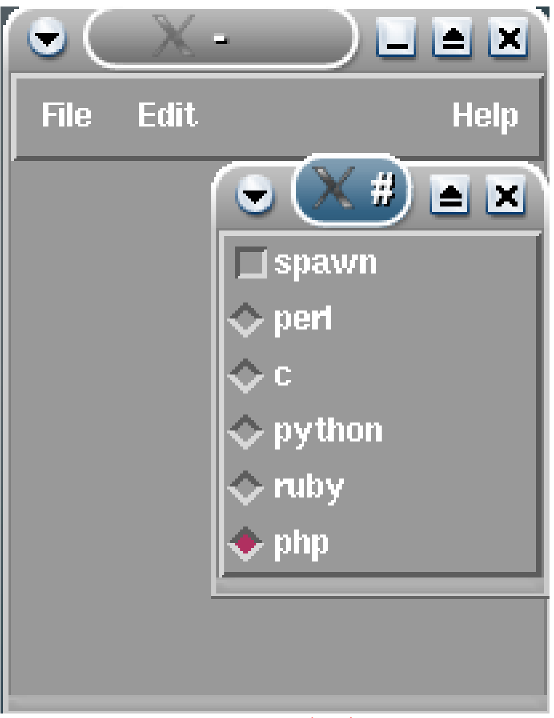
هذا الجزء هو الذي يكون مسئول عن اعطاء الاسم للقائمة

الجزء الثالث
هذا هو الجزء الاهم من اجزاء القائمة وهو الجزء الذي يؤدي الى تكوين القائمة بالشكل المتعارف عليه واليكم الشرح الخاص به

*CODE(130)

```
-menuitems    => [
['checkbutton',"spawn"],
['radiobutton',"perl"],
['radiobutton',"c"],
['radiobutton',"python"],
['radiobutton',"ruby"],
['radiobutton',"php"],
],
```

هذا الجزء يكون مسئول عن تكوين القائمة بالشكل الذي نريده كما ذكرنا نلاحظ ان هذا الخيار اسلوب تعامله مع الاموراو بالاحرى مع الازرار يكون باسلوب مختلف
حيث نلاحظ انه نوع الزر الذي تريد ان تقوم ببرمجته من خلال استعمال هذا الامر تقوم بكتابته بالشكل المكتبوب به في المقطع البرمجي اعلاه و الكلمة النصية التي بجوار نوع الزر الذي قمت باختياره تكون اسم الزر كما هو ملاحظ من خلال الزر وانا قمت باختيار كل من النوعين لكي نعلم انه من الممكن ان تتم العملية ومن الممكن ان تتم اضافة نوعين من الازرار الى نفس القائمة المنسدلة يمكن القول انه الان وبعد ان قرينا هذا المقطع البرمجي انه هذا الكود والبرنامج ليس بالبرنامج الصعب ولكن على العكس هو برنامج جميل ولطيف
وبعد ان نفذنا البرنامج سوف نشاهد صورة البرنامج بعد التنفيذ



*FIGURE(62)

الان وبعد ان قمنا ببرمجة قائمة برنامج وكانت هذه القائمة هي قائمة كما نريد ولكن تبقى تحتاج الى بعض اللمسات

How to Make Advanced Menu with Complex sub Routine

اذا قمنا ببرمجة قائمة وكانت القائمة التي برمجناها لا تعمل شيئا اي انها لاتقوم بعمل محدد فأنه من الافضل ان لانقوم بهذا العمل ونكلف انفسنا عنا ء التعب لكتابة برنامج يبلغ طوله عشرات الخطوات بدون اي فائدة محددة ولذا اذا كنت تريد ان تكتب برنامج خاص لقائمة مثل القوائم التي نراها اصبح لابد ان نعمل برامج اكثر تعقيدا لكي تقوم بوظائف اكثر اهمية مثل خاصية الاستعراض اي خاصية
عمل(browse)
الان سوف نأخذ البرنامج الخاص بعمل ال(browse)
ومن ثم نعمل على دمجه مع البرنامج الخاص للقائمة على كل الان اليكم الكود الخاص بعمل الاستعراض

CODE(131)

```
use Tk;
use Cwd;

$top =MainWindow->new ;

print
   "Filename: ",
   $top->getOpenFile(-defaultextension => ".pl",
                         -filetypes       =>
                         [['Perl Scripts',     '.pl'           ],
                          ['Text Files',       ['.txt', '.text']],
                          ['C Source Files',   '.c',      'TEXT'],
                          ['GIF Files',        '.gif',         ],
                          ['GIF Files',        '',        'GIFF'],
                          ['All Files',        '*',            ],
                         ],
                         -initialdir      => Cwd::cwd(),
                         -initialfile     => "Choose ur file Now",
                         -title           => "Your customized title",
                        ),
   "\n";
```

هذا البرنامج كله وبصورة عامة يمر علينا للمرة الاولى لذا سوف نشرحه بصورة موسعة قليلا
الفقرة الاولى

CODE(132)

```
use Cwd;
```

هذه الفقرة من المهم ان تعرف اولا انها فقرة ليست ملزم بوضعها ولكن من الافضل ان تضعها

HINT

فقرة عليك دائما ان تتذكرها هي انه كل مرة في لغة البيرل تلاحظ وجود كلمة(use)

عليك ان تعرف ان الكلمة التي تأتي من بعد هذه الكلمة هي استخدام للموديل

/EnD/

وفي حالتنا هذه في هذا البرنامج تم استعمال الموديل(Cwd)
والذي يعني عند استعماله(current working directory)
اي في حالتنا هذه البرنامج الذي نكتبه هو يقوم بعمل استعراض للملفات التي موجودة لديك ومع استعمال هذا الموديل سوف تحصل على استعراض للملفات للمكان التي انت فيها موجود

الفقرة الثانية

*CODE(133)

```
$top->getOpenFile(-defaultextension => ".pl",
```

الامر هذا هو امر جديد لم يمر علينا من قبل ولكن على كل هو الامر الذي يكون مسئول عن استعراض الملفات
وفي الخيار الاول المرفق معه نلاحظ انه تم اختيار الامتداد الافتراضي هو الامتداد الخاص بلغة البيرل
وعلى كل هذا الخيار ليس بالخيار المهم في هذا البرنامج اي من الممكن ان يتم الاستغناء عنه
الفقرة الثالثة

*CODE(134)

```
-filetypes      =>
                    [['Perl Scripts',     '.pl'           ],
                     ['Text Files',       ['.txt', '.text']],
                     ['C Source Files',  '.c',      'TEXT'],
                     ['GIF Files',        '.gif',         ],
                     ['GIF Files',        '',        'GIFF'],
                     ['All Files',        '*',            ],
                    ],
```

هذا الخيار صحيح هو خيار طويل ولكن هو خيار بسيط نلاحظ فيه انه تم اختيار امتدادات الملفات المرغوبة لكي يتم استعراضها
وكما هو تلاحظ قد قمت بأختيار
الملفات الخاصة بلغة البيرل
والملفات الخاصة بلغة السي
والصور ذات الامتداد(gif,GIFF)
الى اخره من امتدادات التي من الممكن ان تضيف او تنقص عليها كما تريد
الفقرة الرابعة

*CODE(135)

```
-initialdir     => Cwd::cwd(),
                    -initialfile      => "Choose ur file Now",
                    -title           => "Your customized title",
                   ),
```

الخيار الاول

*CODE(136)

```
-initialdir      => Cwd::cwd(),
```

هو خيار خاص بالاستعراض في ال (*current Working directory*)
والخيار الثاني

*CODE(137)

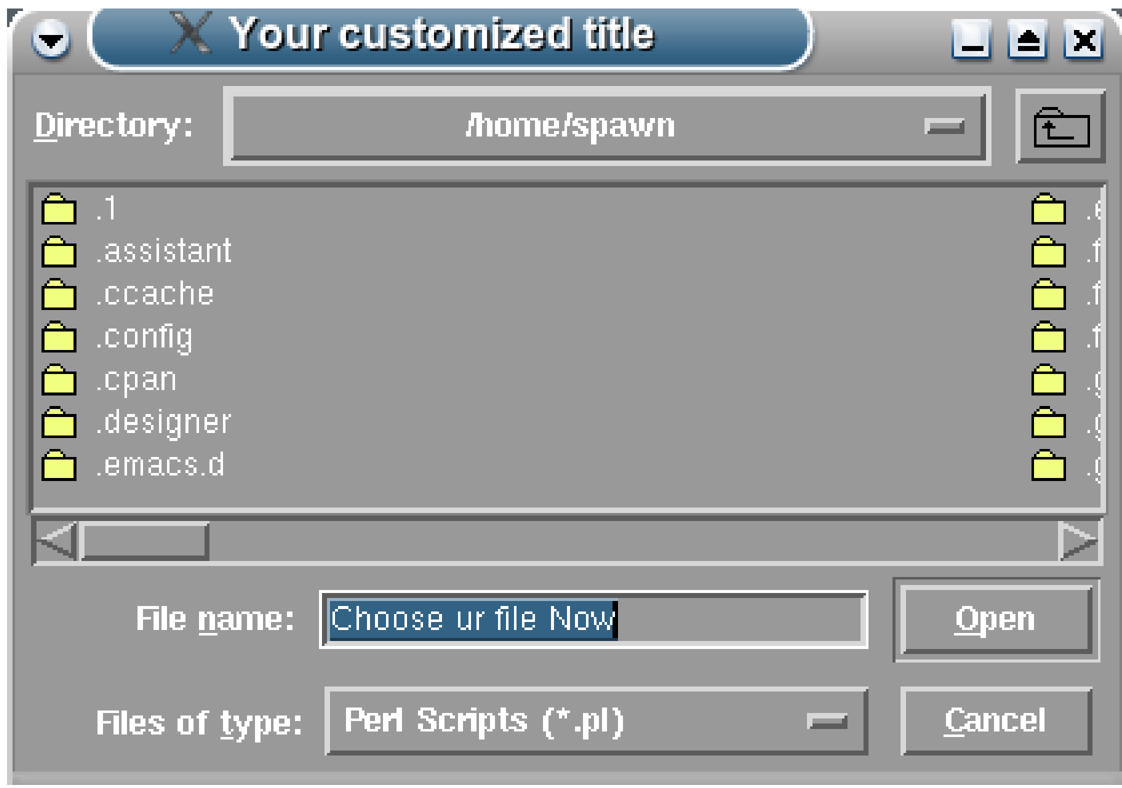
```
    -initialfile      => "Choose ur file Now",
```

هذا الخيار يكون مسئول عن طباعة هذه الجملة في حقل ال(*file name*)
بجوار الزر الخاص بعملية الفتح(*open*)
اما الخيار الثالث

*CODE(138)

```
-title          => "Your customized title",
```

هو مسئول عن اعطاء الاسم للبرنامج
واما الان بعد ان انتهينا من كتابة البرنامج وشرح عملة وفهمه حان الوقت لكي ننفذ البرنامج ونرى ناتج التنفيذ
وهو كما في الصورة الاتية



*FIGURE(63)

نلاحظ خذه هي صورة البرنامج الناتج ولاحظ انه فعلا برنامج مفيد وانه يحتوي ويستعرض جميع الملفات الموجودة في المسار الذي انت فيه ويعرض الملفات سواء كانتمخفية ام غير مخفية
الان سوف نعمل على دمج البرنامج مع البرنامج السابق الخاص بعملية تكوين قائمة و اليكم كيف تتم العملية كما في هذه الكود

***CODE(139)**

```
use Tk;
$top=MainWindow->new;
$menubar=$top->Menu();
$top->configure(-menu        => $menubar);
$file=$menubar->cascade(-label            =>"File");
$edit=$menubar->cascade(-label            =>"Edit");
$help=$menubar->cascade(-label            =>"Help");
$open=$file->command(
                        -label          =>"Open",
                        -accelerator    =>"Ctrl-o",
                        -underline      =>0,
                        -command        =>\&open,
                        );
$file->separator;
$new=$file->cascade(
                        -label          =>"New",
                        -menuitems      => [
['checkbutton',"spawn"],
['radiobutton',"perl"],
['radiobutton',"c"],
['radiobutton',"python"],
['radiobutton',"ruby"],
['radiobutton',"php"],
],
);
$file->separator;
$close=$file->command(
                        -label          =>"Exit",
                        -accelerator    =>"Ctrl-e",
                        -underline      =>0,
                        -command        =>sub {exit},
                        );
$undo=$edit->command(
                        -label          =>"Undo",
                        -accelerator    =>"Ctrl-z",
                        -underline      =>0,
                        );
$edit->separator;
$redo=$edit->command(
                        -label          =>"Redo",
                        -accelerator    =>"Ctrl-y",
                        -underline      =>0,
                        );
$help->command(-label => 'Version', -command => sub {print "This is the first edition of this
programm\n"});
```

```
$help->separator;
$help->command(-label => 'About',   -command => sub {print "this is a simple perl menu
programm\n"});
sub open {
"Filename: ",
   $top->getOpenFile(-defaultextension => ".pl",
                         -filetypes        =>
                         [['Perl Scripts',     '.pl'            ],
                          ['Text Files',       ['.txt', '.text']],
                          ['C Source Files',   '.c',      'TEXT'],
                          ['GIF Files',        '.gif',         ],
                          ['GIF Files',        '',        'GIFF'],
                          ['All Files',        '*',             ],
                         ],
                         -initialdir       => Cwd::cwd(),
                         -initialfile      => "coose ur file now",
                         -title            => "Your customized title",
                        ),
   "\n";

}
MainLoop;
```

الان هذا الكود نلاحظ ان الذي قمنا بعمله فيه هو انه اعتبرنا البرنامج الخاص بفتح المفلفات هو روتين فرعي و اسندنا قيمته الى الخيار(command)
ونلاحظ انه عند تنفيذ البرنامج سوف نحصل على قائمة ولها القدرة على استعراض الملفات الموجودة في الحاسبة
اذن من خلال هذا البرنامج خلصنا الى قاعدة هذه القاعدة هي انه من الممكن ان يتم دمج اي برنامج مع البرنامج الخاص بتكوين القائمة واعتبار البرنامج الذي سيتم دمجه مع البرنامج على انه روتين فرعي يودي وظيفة اضافية من شأن هذه الوظيفة ان تزيد من مرونة وفعالية القائمة

popup menu

لو نرجع بهذه الصفحات قليلا الى الوراء نلاحظ انه كنا قد تعملنا تقنية تدعى بتقنية ال(*top level*)
نلاحظ ان اهم ما يميز هذه التقنية عن غيرها من التقنيات انه من الممكن ان يتم استدعاء نافذة من خلال نافذة
اخرى اي انه من خلال هذه التقنية يتم استدعاء نافذتان من هذه التقنية ولكن من خلال
تقنية ال *popup menu*)
هذه التقنية الخاصة ببرمجة القوائم من خلالها سوف نقوم باستدعاء عدد اكبر من النوافذ وهكذا سوف نحصل على
طريقة اخرى من برمجة القوائم
الان نأخذ الكود الخاص بهذه العملية

*CODE(140)

```
use Tk;
$top=MainWindow->new;
require Tk::Dialog;
my(@popup_opts) = (-popover => undef, qw/-overanchor sw -popanchor sw/);

my $d1 = $top->Dialog(
   @popup_opts,
   -text => "Original options:\n" . join(' ', say(@popup_opts)) .
          "This Dialog should be in the screen's lower-left " .
          "corner.  When you dismiss this Dialog another will " .
          "popup in the southeast corner.",
);
$d1->Show;

@popup_opts = qw/-overanchor se -popanchor  se/;
$d1->configure(
   @popup_opts,
   -text => "Changed options:\n" . join(' ', say(@popup_opts)) .
          "1 second after you dismiss this Dialog another " .
          "will popup, without window manager decorations ".
          "(overrideredirect on), with its southeast corner " .
          "over the cursor.",
);
$d1->Show;
$top->after(1000);

@popup_opts = qw/-popover cursor -popanchor se/;
$d1->configure(
   @popup_opts,
   -text => "Changed options:\n" . join(' ', say(@popup_opts)) .
          "1 second after you dismiss this Dialog another " .
          "will popup, with window manager decorations once ".
          "again , with its northwest " .
          "corner over the cursor.",
);
$d1->overrideredirect(1);
$d1->Show;

@popup_opts = qw/-popanchor  nw/;
$d1->configure(
   @popup_opts,
   -text => "Changed options:\n" . join(' ', say(@popup_opts)) .
          "End of demonstration.",
);
```

```
$d1->overrideredirect(0);
$top->after(1000);
$d1->Show;

sub say {
   map {defined($_) ? $_ : 'undef'} (@_, "\n\n");
}
MainLoop;
```

الان نأتي الى شرح هذا البرنامج وشرح الفقرات المكونة له
الفقرة الاولى

CODE(141)

```
my(@popup_opts) = (-popover => undef, qw/-overanchor sw -popanchor sw/);
```

هذه الفقرة هي عبارة عن فقرة لمصفوفة عادية يحتوي على خيارين من خيارات قائمة ال(*popupmenu*)
وكما نلاحظ من هذه الخيارات هي خيارات خاصة بالموقع الذي سوف تظهر فيه النوافذ كما سنرى عند تنفيذ البرنامج
الفقرة الثانية

CODE(142)

```
my $d1 = $top->Dialog(
   @popup_opts,
   -text => "Original options:\n" . join(' ', say(@popup_opts)) .
           "This Dialog should be in the screen's lower-left " .
           "corner.  When you dismiss this Dialog another will " .
           "popup in the southeast corner.",
);
$d1->Show;
```

الفقرة هذه سوف تكون اول(popup menu)
تظهر لدينا في البرنامج ونلاحظ من هذه الفقرة انه تم استعمال تقنية ال(Dialog)
هي تقنية جديدة و الفائدة من هذه التقنية انه تعمل على عرض النص الذي قمنا بكتابته في الاعلى
اما الجزء الذي يحمل الاسم(say)
فهذا روتين فرعي تم اسناده لكي تتم عملية العرض بالشكل المطلوب
والخيار هذا

CODE(143)

```
$d1->Show;
```

الفائدة منه هو ان تتم عملية الاستعراض

***HINT**

الامر(*show*)
الذي استعملناه الان هو امر يختلف عن الامر(*show*)
الذي استعملناه مع تقنية ال(*entry*)
حيث لا يوجد ربط بين كل من التقنيتين لذا ارجو الانتباه الى هذه الفقرة

الفقرة الثانية

CODE(145)

```
@popup_opts = qw/-overanchor se -popanchor  se/;
$d1->configure(
   @popup_opts,
   -text => "Changed options:\n" . join(' ', say(@popup_opts)) .
           "1 second after you dismiss this Dialog another " .
           "will popup, without window manager decorations ".
           "(overrideredirect on), with its southeast corner " .
           "over the cursor.",
);
$d1->Show;
$top->after(1000);
```

المقطع الثاني من هذا البرنامج هو يعني النافذة الثانية من التي سوف تظهر من البرنامج اي يعني انها ال(nember 2 popup menu)
وهذا الجزء منم البرنامج يتكون مما يلي

CODE(146)

```
@popup_opts = qw/-overanchor se -popanchor  se/;
```

هذا السطر البرمجي يحتوي على اعدادات الاتجاه اي انه في اي مكان على(number 2 popup menu) ان تظهر ومن ثم استعمال خيار الاظهار ولكن الامر هذا

CODE(147)

```
$top->after(1000);
```

هذا الخيار على العموم تقريبا اول مرة نقوم باستعماله وهو يعني بعد كم من الوقت سوف تظهر ال (number 3 popup menu)
اما بالنسبة الى الرقم 1000 فهو يعني انه بعد انقضاء هذا الوقت المحدد وهو 100 سوف تظهر (number 3 popup menu)
الجديدة واما بالنسبة الى معيار قياس الوقت هو(Millseconds)
ومن الممكن ان تغير الرقم هذا الى اي رقم تريد او من الممكن ان تلغيه وعلى راحتك

الفقرة الثالثة

*CODE(148)

```
@popup_opts = qw/-popover cursor -popanchor se/;
$d1->configure(
   @popup_opts,
   -text => "Changed options:\n" . join(' ', say(@popup_opts)) .
          "1 second after you dismiss this Dialog another " .
          "will popup, with window manager decorations once ".
          "again , with its northwest " .
          "corner over the cursor.",
);
$d1->overrideredirect(1);
$d1->Show;
```

هذا المقطع البرمجي من البرنامج يعمل على عرض ال(number 3 popup menu)
ولكن هذا المقطع البرمجي له خاصية مميزة هذه الخاصية المميزة سوف نكتشفها بعد ان يتم تنفيذ البرنامج

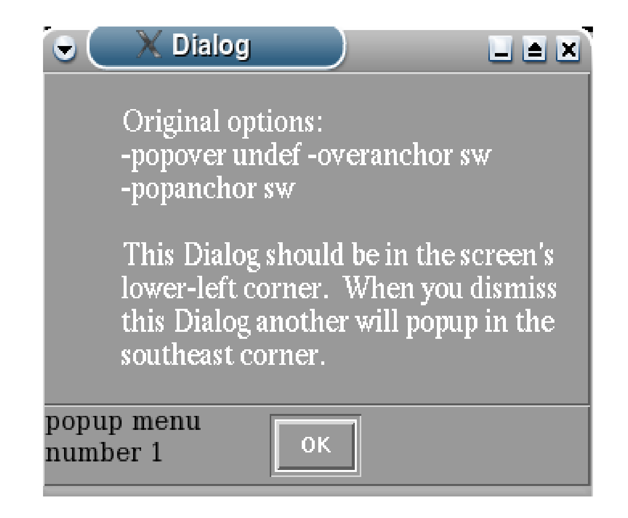
الفقرة الرابعة

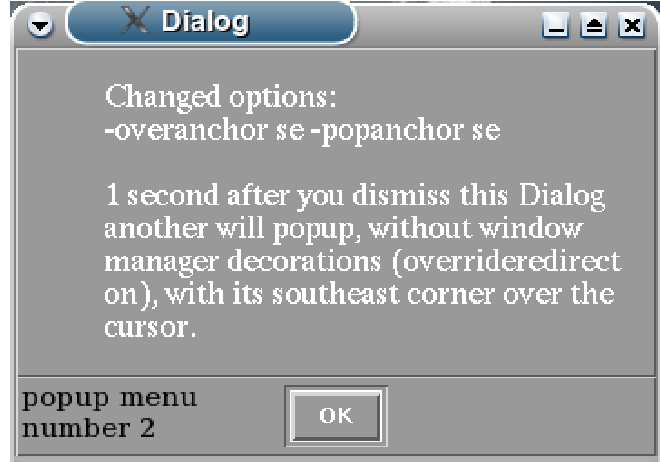
*CODE(149)

```
@popup_opts = qw/-popanchor  nw/;
$d1->configure(
   @popup_opts,
   -text => "Changed options:\n" . join(' ', say(@popup_opts)) .
          "End of demonstration.",
);

$d1->overrideredirect(0);
$top->after(1000);
$d1->Show;
```

هذه الفقرة الرابعة من البرنامج وهي تحدد النافذة الرابعة او ال(nember 4 popup menu)
هذه الفقرة اذا لاحظنا مكوناتها نلاحظ انها لا تحتوي على اي شئ جديد على كل الان سنرى ناتج التنفيذ الخاص بالبرنامج لكي نعلم مالذي جرى من البرنامج

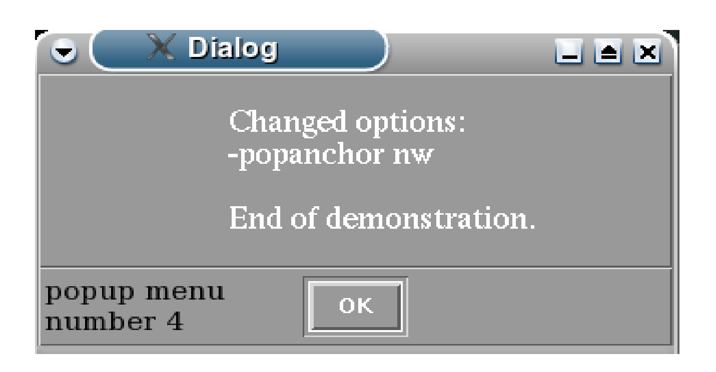




Changed options:
-popover cursor -popanchor se

1 second after you dismiss this Dialog another will popup, with window manager decorations once again , with its northwest corner over the cursor.

popup menu number 3 OK



*FIGURE(64,&,65,&,66,67)

لاحظ الشكل الذي يحمل الرقم 3 فعلا انه شكل يجلب الانتباه لانه لو دققنا الانتباه انه لايحتوى على هذا الشريط في الاعلى منه



*FIGURE(68)

والخيار او الامر الذي يكون مسئول عن اظهار او اخفاء هذا الشريط من البرنامج هو الخيار الاتي

*CODE(150)

```
$d1->overrideredirect(1);
```

هذا هو الخيار المسئول عن هذه العملية

*HINT

في هذا البرنامج لاحظنا استعمال عدد من الخيارات التي علمنا انها ذات علاقة باعدادات الاتجاه و هي

1-

*CODE(151)

```
popover
```

2-

*CODE(152)

```
overanchor
```

3-

*CODE(153)

```
popanchor
```

هذه هي الخيارات التي لها علاقة باعدادات الاتجاهات في برمجة الواجهات الرسومية لكن ما احب ان اوضحه من القيم التي تسند الى هذه الخيارات هو الخيار الذي يحمل الاسم

*CODE(154)

```
cursor
```

حيث ان هذا الخيار له استراتيجية معينة في العمل حيث هنالك خيارات لها سياسة محددة في مكان اظهار النوافذ ولكن هذا الخيار يعتمد على اظهار النوافذ اعتمادا على موقع ال(Cursor of the mouse)
اي اينما يكون مؤشر الماوس هو يظهر النافذة هنالك لذا من الممكن ان تسبب هذه الفقرة تشوش عند بعض الاشخاص لذا يجب الانتباه الى مثل هذه الحركات

Option Menu

هذا النوع من القوائم يعتبر من الانواع البسيطة من القوائم و التي كثير منا يصادفها في حياته وخاصة في مواقع الانترنت على كل هذه القوائم ليست بطول القوائم التي سبق و ان اخذناها في الصفحات القليلة الماضية و ان اسلوب عملها يتم كما يلي من خلال هذا البرنامج

***CODE(155)**

```
use Tk;
$top=MainWindow->new;
$b=$top->Label(
            -text           =>"Choose ur language",
            -foreground     =>'red',
            );
$b->pack;

@a=(qw /perl c python ruby java cpp c# php asmebly coldfusion bashscript Vb /);
$c=$top->Optionmenu(
            -options                =>[@a],
            -command                =>sub {print "Ur favourite is:-@_,\n",},
);
$c->pack;
$d=$top->Button(
            -text           =>"Exit",
            -command        =>sub {exit}
            );
$d->pack(-side        =>'bottom');
MainLoop;
```

على كل لو تلاحظ الكود وتمعن النظر فيه لسوف تلاحظ ان جميع مكونات الكود قد مرت علينا وجميع مكوناته نعلم ما هو عملها وقبل ان نشرح عمل الكود اليكم الصورة الخاصة بهذا البرنامج



*FIGURE(69)

الان نأتي الى شرح البرنامج و الفقرة الوحيدة التي سوف نشرح عنها هي الفقرة التالية

*CODE(156)

```
@a=(qw /perl c python ruby java cpp c# php asmebly coldfusion bashscript Vb /);
$c=$top->Optionmenu(
                -options                =>[@a],
                -command                =>sub {print "Ur favourite is:-@_,\n",},
);
```

السطر الاول

*CODE(157)

```
@a=(qw /perl c python ruby java cpp c# php asmebly coldfusion bashscript Vb /);
```

عبارة عن مصفوفة وهذه المصوفة تم ادخال اسماء اللغات على انها العناصر الخاصة بها

السطر الثاني

*CODE(158)

```
$c=$top->Optionmenu(
```

هذا السطر يكون مسئول عن تكوين القائمة التي سوف تكون قائمة الخيارات

السطر الاول من البلوك البرمجي

*CODE(159)

```
        -options                =>[@a],
```

السطر يكون عن ادخال اسماء اللغات الموجودة في المصفوفة التي في الاعلى الى القائمة

السطر الثاني من البلوك البرمجي

*CODE(160)

```
-command                =>sub {print "Ur favourite is:-@_,\n",},
```

هذا السطر كما تعودنا من الامر كوماند يتم اسناد روتين فرعي اليه في برنامجنا هذا قمنا باستخدام الجملة الطباعة في روتين فرعي وكان عمل جملة الطباعة هذه هي ان تقوم بطباعة جملة

ان لغتك المفضلة هي واسم اللغة طبعا بما انه قلنا جملة الطباعة اذن فأن ناتج تنفيذ الروتين الفرعي هذا سيتم تنفيذه في الشيل

Tix Widget

في برمجة الواجهات الرسويمةلا يوجد تقنية تحمل هذا الاسم وان هذه التقنية تكون عبارة عن اختصار ل(*Tk interface Extesions*)
يعني لن نشاهد برنامج يتم استعمال هذه التقنية معه ولكن هذه التقنية تضم تحتها تقنيات اخرى يتم التعامل معها
ولكن ما سبب استعمال هذا الاسلوب حقيقة لا ادري لماذا
والتقنيات المنضمة تحت جناح الى هذه التقنية هي

1-TList
2-HList
3-Dirtree

1- Tlist

وهذه التقنية تشبه تقنية ال(*Listbox*)
ولكن هي تقنية مرنة اكثر امثر من تقنية ال(*listbox*)

2-HList

وهذه التقنية هي هي مختصر لكلمة(*hierarchical list*)
اي اذا ما ترجمت الى العربية تعني القائمة المرتبة وهذه التقنية غالبا ما يتم استعمالها على انها (*Fancy listbox*)

3-Dirtree

هذه التقنية يتم استعمالها لكي تقوم بعرض المجلدات بشكل(*hierarchical list*)
اي نوعا ما فب طريقة عرضها تشبه ال(*HList*)
كما سنلاحظ بعد قليل وهي تقنية خاصة لعرض المجلد كما هو واضح من اسمها

Simple using of TList

1-how to make ur First Program

بعد ان قمنا بعمل تعريف بسيط لهذه التقنية الان سوف نقوم باستخدامها في البرامج لكي نرى كيف يتم عملها اليكم الكود الاول الخاص بهذه التقنية

*CODE(161)

```
use Tk;
$top=MainWindow->new(-title      =>"TList");
require Tk::TList;
$a=$top->TList(-width       =>40);
foreach (qw /perl c python ruby java cpp c# php asmebly coldfusion bashscript Vb /){
$a->insert('end',
                    -text             =>$_,
                    -itemtype         =>'text'
          );
}
$a->pack;
MainLoop;
```

الان لو قمنا بتنفيذ الكود سوف نحصل على الصورة الاتية من ناتج التنفيذ



*FIGURE(70)

الان لو تلاحظ الكود هذا سوف ترى ان الفقرة التي لم تمر علينا هي هي الفقرة التالية

*CODE(162)

```
$a=$top->TList(-width       =>40);
```

و هي كما ذكرنا الفقرة هي المسئولة عن تكوين المنصة الرئيسية الخاصة بال(TList)
اما الخيار الاخر الذي لم يمر علينا هو الخيار التالي

*CODE(163)

```
-itemtype      =>'text'
```

اما هذا الخيار يكون مسئول عن اخبار مترجم البيرل الخاص ببرمجية الواجهات الرسومية عن نوع ال(items)
الذي سوف يتم التعامل معه و هو كما نلاحظ في حالتنا هذه قمنا باخباره ان النمط الذي نريده هو ان يكون نص

Advanced Using of the TList

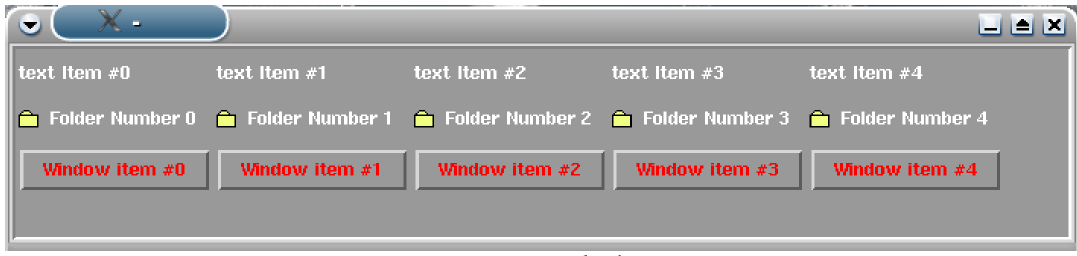
في هذا الاستعمال الثاني لهذه التقنية سوف نقوم بتكوين(TList)
اكبر واوسع واكثر شمولية حيث سوف نقوم بدمج النصوص مع الصور ومع الازرار وسوف تتم هذه العملية كما في الكود الاتي

CODE(164)

```
use Tk;
$top=MainWindow->new;
require Tk::TList;
$tl = $top->TList(-width      =>100)->pack(-expand => 1, -fill => 'both');
my $image = $top->Getimage('folder');

foreach my $i (0..4) {
  $tl->insert('end', -itemtype => 'text', -text => "text Item #$i");
  $tl->insert('end', -itemtype => 'imagetext',
            -text => "Folder Number $i", -image => $image);
  my $b = $tl->Button(          -text => "Window item #$i",
                          -foreground    =>'red',
                          -command => sub { print "Button pressed,$i\n"; });
  $tl->insert('end', -itemtype => 'window', -widget => $b);
}
MainLoop;
```

الان عند تنفيذ هذا الكود سوف نحصل على الصورة التالية و هي كما يلي



FIGURE(71)

هذه هي صورة البرنامج الان نأتي الى شرح المكونات البرمجية الخاصة بالبرنامج و هي كما يلي
الفقرة الاولى

CODE(165)

```
$tl = $top->TList(-width      =>100)->pack(-expand => 1, -fill => 'both');
```

هذا الجزء من الكود يكون مسئول عن تكوين المنصة الاساسية الخاصة بال(TList)
وخصصنا لهذه المنصة اي المنصة الخاصة بال(TList)
فقط ان يكون مقدار عرضها حوالي ال 100
الفقرة الثانية

CODE(166)

```
my $image = $top->Getimage('folder');
```

هذه الفقرة هي الفقرة الخاصة بعمل استيراد او جلب الصورة الخاصة بصورة المجلد من الحاسبة ولكن يجب ان

تعرف ان ليس بمقدورك ان تجلب اي صورة ترغب بها لانه هذه التقنية تعمل على استيراد الصور من المسار الافتراضي الذي يتم تنصيب الموديل الخاص ببرمجة الواجهات الرسومية يعني بعبارة اخرى
المكان الذي قمت بتنصيب الموديل الخاص ببرمجة الواجهات الرسومية فيه هو يعتبر المسار الافتراضي الذي يتم فيه وهو غالبا يكون المسار التالي

*CODE(167)

```
/usr/lib/perl5/site_perl/5.8.8/i386-linux-thread-multi/Tk
```

ولو فتحت هذا المسار او رأيت الصور التي في داخله هذه الصور هي الصور التي بامكانك ان توردها كما تريد
الفقرة الثالثة

*CODE(168)

```
foreach my $i (0..4) {
```

هنا هذه الفقرة البسيطة تحتوي على جملة التكرار من صفر الى اربعة وفقط
الفقرة الرابعة

*CODE(169)

```
$tl->insert('end', -itemtype => 'text', -text => "text Item #$i");
```

هنا في هذه الفقرة
من خلال استخدام الامر الخاص بالادخال قمنا بادخال العبارة الواضحة لكي تظهر لدينا العبارة التي نريد لكي تظهر في اول سطر من البرنامج
الفقرة الخامسة

*CODE(170)

```
$tl->insert('end', -itemtype => 'imagetext',
        -text => "Folder Number $i", -image => $image);
```

في هذه الفقرة نقوم بضبط الاعدادات الخاصة بالصورة التي نعمل على استيرادها ونلاحظ هنا في ذا الكود قمنا بتغير الخيار(itemtype)
من الاسلوب النصي الذي استعملناه في البرامج السابقة الى الاسلوب هذا

*CODE(171)

```
Itemtext=>'imagetext',
```

لكي نحصل على نتيجة افضل
ومن ثم نختار النص الذي نريد ان يكون مطبوع الى جوار الصورة التي نريدها ومن ثم من خلال استخدام الامر الامر

*CODE(172)

```
-image=>$image
```

يتم اعتماد استيراد الصورة
الفقرة السادسة

*CODE(173)

```
my $b = $tl->Button(        -text => "Window item #$i",
                    -foreground     =>'red',
                    -command => sub { print "Button pressed,$i\n"; });
```

هذه الفقرة هي الفقرة المسئولة عن تكوين الازرار في البرنامج ونلاحظ ان جميع مكوناتها بسيطة لذا لاتحتاج الى شرح فقط هنالك فقرة احب ان اشير اليها هي في البلوك البرمجي الخاص بالزر وهو الخيار

*CODE(174)

```
-command => sub { print "Button pressed,$i\n"; });
```

هنا هذا الخيار نعرف اه مسنود الى روتين فرعي ولكن في جملة الطباعة سبب اضافة المتغير($i)
الى الجملة هو حتى عندما تضغط على الزر يقوم بكتابة تسلسل الزر الذي ضغطت عليه يعني عندما تضغط على الزر رقم 2 فأنه يكتب لقد ضغطت على الزر 2

الفقرة السابعة

*CODE(175)

```
$tl->insert('end', -itemtype => 'window', -widget => $b);
```

هذه الفقرة يكون لها اهمية فيعرض النافذة التي برمجنا فيها المقطع الخاص ببرمجة الازرار فلولا وجود هذه النافذة لما كان من الممكن ان يتم عرض المنصة الخاصة بالازرار
حيث نلاحظ في هذه القفرة انه لايوجد الامر الخاص بالحزم اي انه لايوجد سطر برمجي مثل هذا

*CODE(176)

```
$ tl->pack;
```

لماذا لاوجد امر مثل هذا ولماذا في هذا النوع من التقنية تم الاستغناء عنه الان سوف نرى صورة لنفس البرنامج ماذا سوف يحدث له فيما لو تم الاستغناء عن الفقرة الاخيرة التي تكلما عنها
الان هنالك سؤال يطرح و هو
لماذا تم كتابة هذه الخطوة بالشكل التالي

*CODE(177)

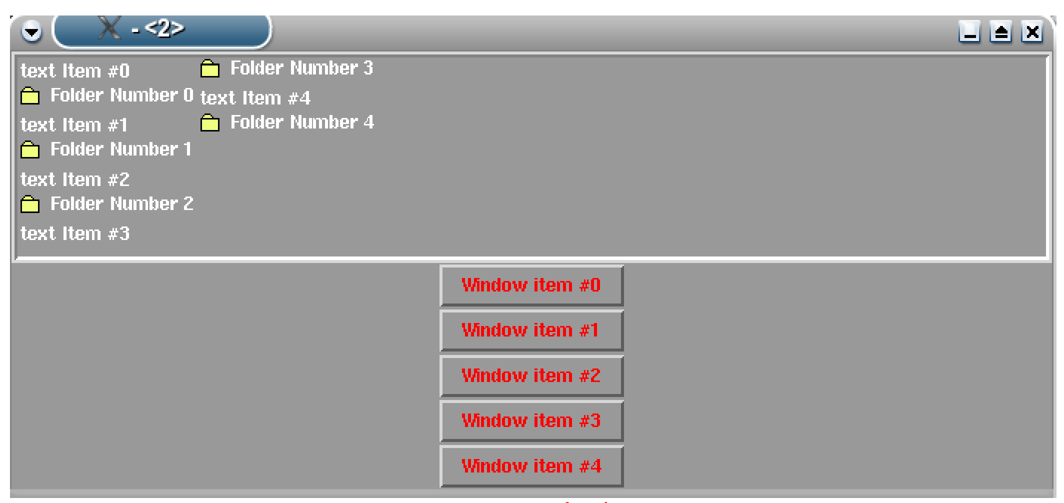
```
my $b = $tl->Button( -text => "Window item #$i",
                      -foreground    =>'red',
                      -command => sub { print "Button pressed,$i\n"; });
```

لماذا كتب بهذا الشكل؟؟ ولماذا لم يكتب بهذه الصورة

*CODE(178)

```
my $b = $top->Button(         -text => "Window item #$i",
                      -foreground    =>'red',
                      -command => sub { print "Button pressed,$i\n"; });
  $b->pack;
```

المقطع هذا هو من الناحية البرمجية هو صحيح ولكن من الناحية التنفيذية هو غير مقبول الان سوف نرى صورة البرنامج فيما لو كانت الفقرة اعلاه هي التي تم اعتمادها في البرنامج بدلا من تلك الاصلية



*FIGURE(72)

نلاحظ في هذا البرنامج انه يبدو وكأنه يحتوي على نافذتين ولكن كما ذكرنا انه لايوجد خطأ برمجي ولكن هو خلل من الناحية الكمالية لذا كان لابد من جود الخطوة التي ذكرناها

ان اهمية الخيار

*CODE(179)

```
-widget-> $b
```

ان هذا الخيار من خلاله نقوم باختيار النافذة التي نريد عرضها ولكن من مميزات هذا الخيار هو انه مرادف للخيار

*CODE(180)

```
-window
```

اي من الممكن ان يتم التبديل بينهما من دون الحصول على خلل و الحصول على نفس الناتج ولك ان تجرب ذلك

HList

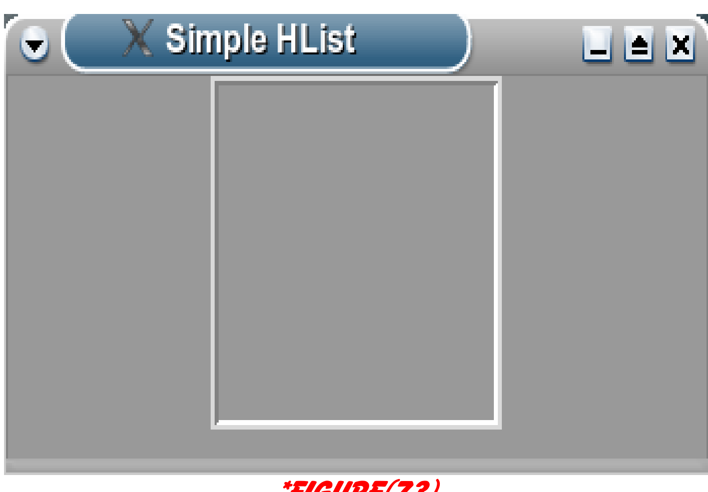
هذه التقنية هي التقنية الثانية من تقنيات ال(TIX)
وهي تعتبر الاساس المهم والاساسي في برمجة ال(Dirtee)
كما سوف نرى

الان سوف نأخذ برنامج بسيط يوضح عمل ال(HList)

*CODE(181)

```
use Tk;
require Tk::HList;
$top=MainWindow->new(-title          =>"Simple HList");
$a=$top->HList()->pack;
MainLoop;
```

هذا هو ابسط كود خاص لتكوين ال(HList)
وان الناتج من تنفيذ هذا الكود كما في الصورة التالية



*FIGURE(73)

Adding text to the Hlist

اغلب التقنيات السابقة مثل

1-listbox
2-HList
3-Text
4-tagconfigure
5-Scrolled
6-scrollbar

وغيرها من هذه التقنيات لو كنا نريد ان نقوم بادخال النص اليها كنا نقوم بذلك من خلال استخدام الامر

CODE(182)

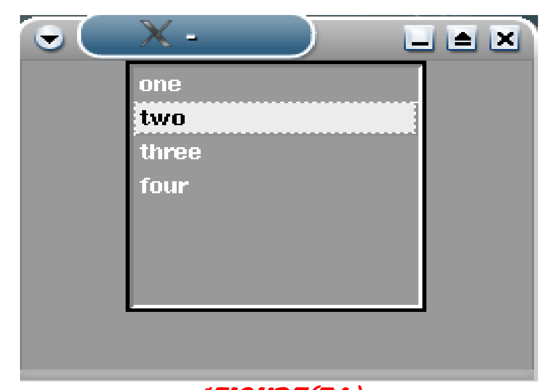
```
insert
```

كان هذا الامر هو الامر المسئول عن ادخال النص الى التقنيات السابقة اما عن هذه التقنية فاسلوب ادخال النص يكون اسلوب مختلف و هو كما يلي في هذا الكود

CODE(183)

```
use Tk;
require Tk::HList;
$top=MainWindow->new;
$a=$top->HList();
foreach (qw/one two three four/){
$a->add($_,
                    -text    =>$_
                    );
}
$a->pack;
MainLoop;
```

الان سوف نرى صورة البرنامج و هي كما يلي



FIGURE(74)

نلاحظ انه في هذا البرنامج تم استعمال اسلوب جديد في ادخال النص و هو من استعمال الامر

*CODE(184)

```
add
```

وهذه اول مرة تمر علينا هذه الخاصية وعلى كل الان سوف نشرح البلوك البرمجي الخاص بها

*CODE(185)

```
$a->add($_,
            -text    =>$_
            );
```

الامر

*CODE(186)

```
add
```

هو كما ذكرنا الامر الذي يكون مسئول عن ادخال النص الى هذه التقنية

*CODE(187)

```
$_,
```

سبب وجود هذا السطر البرمجي هو انه يكون مسئول عن ادخال اول رمز او كلمة او حرف موجودة في عبارة التكرار الى ال(*HList*)

ولك ان تجرب البرنامج الذي كتبناه الان بدون هذا الخيار عند التنفيذ سوف تحصل على خلل برمجي يخبرك بعدم قدرته على ادخال القيمة الاولى

*CODE(188)

```
-text    =>$_
```

اما عن هذا الامر فهو يكون مسئول عن ادخال الرمز او الكلمة او الاحرف الى اخره من الكلمات الموجودة في جملة التكرار الى البرنامج اذن الخيار الذي يسبق هذا يكون مسئول عن ادخال الكلمة او الحرف الاول الى البرنامج او الامر الثاني فهو يكون مسئول عن ادخال الباقي

Scrolling the HList

في بداية شرحنا لتقنية ال(HList)
ذكرنا انها مختصر لكلمة(hierarchical list)
اي القائمة المرتبة و الان حان الوقت لكي نطبق المعني الحرفي لهذه التقنية من خلال استعمال عدد من التقنيات وتتم هذه العملية كما يلي من خلال هذا الكود

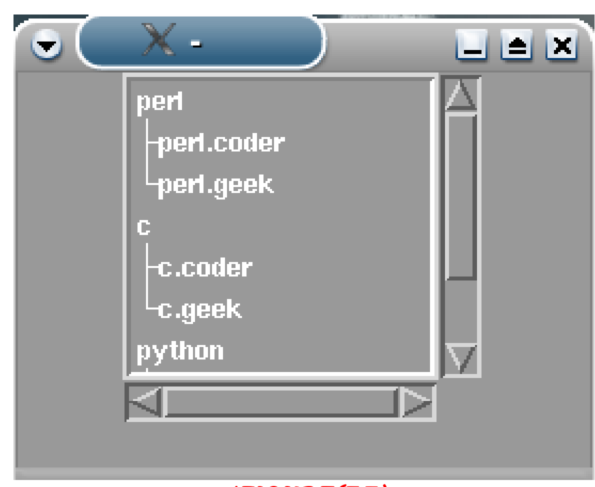
*CODE(189)

```
use Tk;
require Tk::HList;
$top=MainWindow->new;
$a=$top->Scrolled('HList',-scrollbars        =>se
                    );
$a->HList();
foreach(qw/ perl perl.coder perl.geek c c.coder c.geek python python.coder python.geek/){
$a->add($_,
                    -text               =>$_,

          );

}
$a->pack;
MainLoop;
```

الان نأتي الى ناتج تنفيذ البرنامج و هو كما في الصورة التالية



*FIGURE(75)

هذا هو ناتج تنفيذ البرنامج اما الان نأتي الى شرح الفقرات البرمجية المكونة للبرنامج
الفقرة الاولى

*CODE(190)

```
$a=$top->Scrolled('HList',-scrollbars      =>se
                    );
```

هذه الفقرة سبق وان مرت علينا من قبل و هي انه من خلال هذه الفقرة نحدد موضع السكرول بارز في البرنامج

الفقرة الثانية

*CODE(191)

```
$a->HList();
```

هذا الامر هو الامر الذي يكون مسئول عن تكوين المنصة الخاصة بال(HList)
الفقرة الثالثة

*CODE(192)

```
foreach(qw/ perl perl.coder perl.geek c c.coder c.geek python python.coder python.geek/){
$a->add($_,
                -text               =>$_,

        );
```

هذه الفقرة كما نلاحظ مكونة من مقطعين
عبارة الفور مكونة من اسما ء اللغات ومن بعد اسماء اللغات يوجد ملحقات بنأء على على اسماء اللغات وملحقاتها سوية يتم تكوين الشجرة وحتى لو اضفت اكثر من لغة او كل يعنيلو اردت ان تكون الشجر الاولى تكون مكونة من 3 فروع و الاخرة مكونة من فرعين هذا ايضا يجوز لكي تتوضح لديك الصورة عليك ان تعرف ان بنأء الشجر يتم على اسم اللغة حيث هي التي تحدد اي لا يلعب الملحق دورا في عملية الترتيب هذه
واما عن الخيارات التي تأتي مع الامر ال

*CODE(193)

```
add
```

اصبحت الان معروفة لدينا

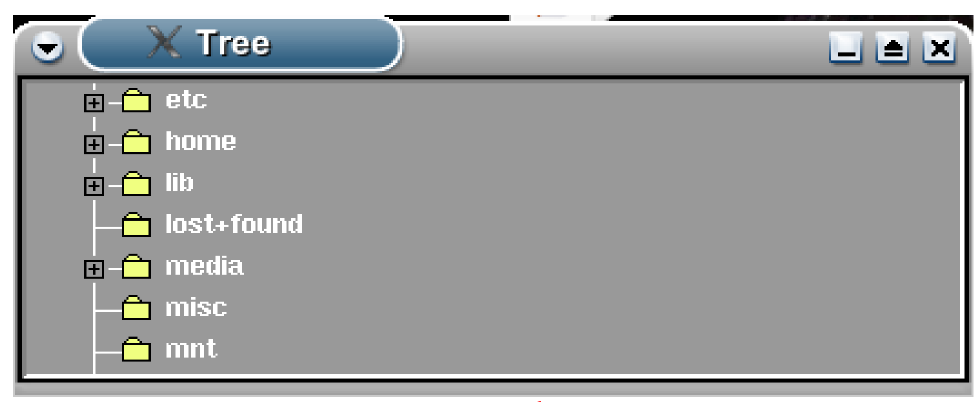
DirTree

هذه التقنية هي التقنية التي ذكرنا انها تقوم بعرض الملجدات التي لديك على الحاسبة على شكل شجرة واسلوب عملها بسيط وهو كما يلي

*CODE(194)

```
use Tk;
use Tk::DirTree;
my $top= MainWindow->new(-title => 'Tree');
$top->DirTree(-directory => "/",
              -width =>30)->pack( -expand => 1);
MainLoop;
```

نلاحظ من هذا الكود ان اسلوب عمله هو بسيط وان الناتج التنفيذ هو كما يلي في الصورة التالية



*FIGURE(76

على الرغم من ان هذه التقنية اول مرة نتعرف عليها الا انها تبدو سهلة جدا الفقرات المكونة لها هي ايضا بسيطة وهي كما يلي

الفقرة الاولى

*CODE(195)

```
$top->DirTree(-directory => "/",
              -width =>30)->pack( -expand => 1);
```

هذه الفقرة من خلالها نحدد الملجد الذي نريد ان يتم استعراضه ونحن هنا في هذا البرنامج حددنا الجذر الرئيسي للنظام ولكن من الممكن ان تحدد غير مسار مثلا ان تحدد

/etc
/usr

الى اخره من المجلدات

ملاحظة من الممكن ان تقوم من خلال هذا البرنامج ان تقوم باستعراض الملفات المخفية في النظام بمجرد ان تقوم بكتابة مسارها اي كما في هو الحالات العادية الاخرى

M_SpAwN Last Touch

*LICENSE

you want to get developed in the Fr34ky life? So What the hell you waitin for check these fr34ky webz now they might help ya see

*BLOGS.

1-
Www.linux-fr34k.com
(Linux security blog but it`s so dangerous like TNT Blog Hush don`t tell any one)

2-
www.binary-zone.com
(Special blog of the G0dF4th3r of Arabic Linux world)

3-
www.snix.wordpress.com
(the blog of the inspired designer)

---

*W3B FORUMS

1-
www.programming-fr34ks.net
(the Aerie of fr34ks check us there and be with us don`t be late)

2-
www.securitygurus.net
(the real way of security our home we all miss you)

here is some useful perl sites
1-
www.perl.org
(official perl site)
2-
www.perl.com
(Rich contents of perl infromation)
3-
www.cpan.org
(Every thing u need for perl modules and the tuts related to these modules)

for more site visit
www.perl.org
and this site well give you quite enough sites

*NOT3(1)
Always remember this tip P3rl is not the strongest programming language but is one of the best programming language at all becoz it`s so easy and flexible lang and this lang have an Extraodinary capabilities and very Advanced F34tures always remeber P3rl is made for Wisdom seekers

*NOT3(2)
اذا احببت لغة البيرل بشكل عام واحببت هذا الموضوع اي برمجة الواجهات الرسومية و اردت ان تتعمق اكثر في هذا المجال البرمجي من الممكن ان تنتظر الملحق الخاص بها الكتاب الي راح يتناول البرمجة المتقدمة للواجهات الرسومية واذا احببت ان تقرأ كتاب اخر اي كتاب انكليزي هنا احب ان اعطيك نصحية هي انه اقرأ ما تحب وطبق ما تحب ولكن ابق بعيدا عن كتاب ال
Advanced perl Programmming
لانه سيضرك اكثر من مما سوف ينفعك هذا من وجهة نظري البسيطة لو كانت تهمك

*NOT3(3)
to the ppl i forgot you were`t in my mind for some reasons you know it and you don`t deserve any thanks for any thing you should know that very well

*GR33TZ
Now After my book is finished i want to say thanks to my all friends who encourage me and always give the help,support,and hope thanks for benig such a good ppl
book wallpaper is designed by MR.(SNIX)
(St0rm_MaN,StrikerX,Snix(N H 2 0 0 4))
M4ILm3:-Mahmoud_najafy@hotmail.com